SISTEMAS OPERATIVOS 2021-1



SANTIAGO ARISTIZABAL CORREA

MANUAL DE TECNICO

CH-MAQUINA

VERSION 3.0

**REQUISITOS TECNICOS**

## **Requerimientos mínimos de Hardware**

* Dado que la aplicación se desarrolló como plataforma web, el único requisito mínimo en cuanto a Hardware es una conexión.

# **DEFINICIÓN DE HERRAMIENTAS USADAS**

La implementación del proyecto CH-MAQUINA fue realizada en el lenguaje de programación JavaScript, además de usar elementos de HTML y CSS.

A continuación, se especifican los elementos usados en el desarrollo de la aplicación.

## **Editores de código**

**Visual Studio Code:** es un editor de código fuente. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado.

1. **Lenguajes de programación:**

**HTML:** Conjunto de etiquetas que permiten generar la interfaz para que el usuario pueda usar la aplicación.

**JavaScript:** Lenguaje de programación usado de lado del cliente para la validación de valores de entrada por parte del usuario por medio de HTML.

**CSS:** Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML y derivados.

1. **Aspectos del programa**

El programa debe simular un procesador muy elemental y una memoria principal a

través de un vector de hasta 9999 posiciones, las cuales pueden ser variadas al

momento de iniciar el programa, se asume por defecto que el chcomputador

empieza con 100 posiciones de memoria para facilitar el proceso de pruebas. El

programa debe estar en capacidad de leer un conjunto de programas en un seudo

lenguaje de máquina que llamaremos CHMAQUINA y los cargará en las

posiciones disponibles de la citada memoria, leerá una instrucción por cada

línea de entrada.

Las primeras posiciones de la memoria estarán reservadas para el núcleo del

sistema operativo (kernel), el tamaño de este deberá poderse ingresar al iniciar la

corrida del simulador, su valor por defecto es 10\*z + 9 posiciones (donde z será el

último digito de la cedula del estudiante).

El programa deberá realizar un chequeo de Sintaxis, produciendo un listado de

errores si los hay, de lo contrario procederá a la carga definitiva del programa en

memoria y quedará listo para ejecución del mismo bajo las reglas de corrida de

múltiples programas

1. **Estructura del proyecto**

Interfaz de usuario gráfica

Descripción generada automáticamente

El proyecto en su estructura tiene diferentes carpetas las cuales almacenas lo diferentes tipos de archivos y códigos realizados en el desarrollo del proyecto.

1. **Paquete CHmaquina app.js**

En este punto se dará una explicación concisa de algunas partes del código encargadas de ciertas funcionalidades.

et memoria = [];

let listaArchivos = []

var numVar=0; //Es contador que cuenta las "VARIABLES" que hay por archivo

let contPasoApaso = 0;

let acumStepbyStep = 0;

let nextFile = 0

var sum = 0;

var arrayVariables = [];

var instrucciones = [];

// De izquierda a derecha en la interfaz

var file = document.getElementById('files');

let btnEncender = document.getElementById('encender');

let btnApagar = document.getElementById('apagar');

let kernel = document.getElementById('kernel');

let memoriaInput = document.getElementById('memoria-input');

let inputAcumulador= document.getElementById('acumulador');

let acomulador = inputAcumulador.value;

var ejecutar = document.getElementById('correrPrograma')//nuevo

let divMemoria = document.getElementById('memoria');

// variables del contenedor footer

let idColumn= document.getElementById('idColumn');

let programa= document.getElementById('programa');

let ins= document.getElementById('ins');

let rb= document.getElementById('rb');

let rlc= document.getElementById('rlc');

let rlp= document.getElementById('rlp');

// no importantes

let monitor= document.getElementById('monitor-result');//diferente

let modo = document.getElementById('modo');

let impresora= document.getElementById('impresora-result');//diferente

let cerrar = document.getElementById('cerrar');

let infoFooter= document.getElementById('footer-home');

let btnStepbyStep=document.getElementById('stepByStep');

//  array de Id

let listId= [];

let numId= 0;

//  array de programa

let listPrograma=[];

let listIns=[];

let listRb=[];

let listRlc=[];

let listRlp=[];

En un inicio se puede encontrar la definicon de las variables que seran utilizadas en la ejecución del sistema con el comando LET, encargado de asignar los valores a estas. Tambien se encuentran declarados los vectores utilizados en el mismo.

**FUNCION ENCENDER()**

function encender() {

    if (Number(memoriaInput.value) > 9999) {

        alert('EL ESPACIO DE MEMORIA HA EXCEDIDO');

        apagar();

    }

    kernel.disabled=true;

    memoriaInput.disabled=true;

    btnEncender.style.display = 'none';

    btnApagar.style.display = 'inline-block';

    modo.innerHTML = 'Modo usuario';

    cerrar.style.display='block';

    files.disabled = false;

    let memoriaMostrar = [];

    let lAcumulador = [0,' Acumulador']

    memoriaMostrar.push(lAcumulador.toString().replaceAll(',',' '));

    // console.log(memoriaMostrar);

    for(let i = 1; i<Number(memoriaInput.value)+1; i++) {

        if(i <= Number(kernel.value)) {

            let so = [i];

            so.push('CHSO\_V2021');

            memoriaMostrar.push(so.toString().replaceAll(',',' '));

        } else{

            memoriaMostrar.push(`${i} - - - - `)

        }

    }

    // console.log(memoriaMostrar);

    document.getElementById('memoria').style.display = 'block';

    document.getElementById('memoria').innerHTML = memoriaMostrar.join('<br></br>');

    // Button style to file

    document.getElementById('file\_icon').style.opacity = '100%';

    document.getElementById('files').style.cursor = "pointer";

}

Esta función se encarga de hacer el arranque de la interfaz, aplica la inicialización del mapa de memoria y desactivación de botones de carga de archivo, encendido de máquina y los medidores de memoria y kernel. Permite garantizar que antes de ejecutar los programas, se pueda demostrar que el valor que hay en memoria, no exceda de 9999 posiciones. Si se cumple con esta condición se crea la lista, con el acumulador, el espacio de memoria (kernel) y las posiciones de memoria restantes. Permitiendo dar entrada a la selección de archivos.

**FUNCIÓN APAGAR ()**

function apagar() {

    location.reload()

    kernel.disabled = false;

    memoriaInput.disabled = false;

    btnApagar.style.display = 'none';

    btnEncender.style.display = 'inline-block';

    modo.innerHTML = 'Modo kernel';

    files.disabled = true;

    btnMemoria.disabled = false;

}

Esta funcion por el contrario de la función encender, activa nuevamente los medidores Kernel y de Memoria permitiendo su modificación, y a su vez cambia el modo del sistema a modo kernel.

**FUNCION LEERARCHIVO()**

filesCH = []

var initialPosition = 0;

let listaEtiquetas = []; //Lista para etiquetas para Mostrar en el div

function leerArchivo(evento) {

  let archivoCH= new ArchivosCH()

    for(let i=0; i<evento.target.files.length; i++) {

    document.getElementById('instrucciones').innerHTML = "";

    document.getElementById('variables').innerHTML = "";

    document.getElementById('etiquetas').innerHTML = "";

    let archivo = evento.target.files[i];

    let name= evento.target.files[i].name;

    archivoCH.ipMemory = initialPosition;

    archivoCH.name = name;

    //GUARDA EL NOMBRE DEL ARCHIVO A CARGAR

    procesarArchivo(archivo, function(result) {

      let lArchivo = []; //LINEAS PERO SIN ID

    lArchivo = result.split('\n');

    //funcion que elimina espacios, comentarios entre otros

      for(let i=0; i<lArchivo.length; i++) {

        if(lArchivo[i] == "") {

          lArchivo.splice(i, 1);

          i--;

        }

        if(lArchivo[i].length == 1) {

          lArchivo.splice(i, 1);

          i--;

        }

        // if(lArchivo[i].includes('//')) {

        //   lArchivo.splice(i,1);

        //   i--;

        // }

      }

    let listaPrueba = [];//LISTA DE LOS ARCHIVOS ORGANIZADOS CON EL ID QUE APARECE EN LA INTERFAZ

    let listaFile=[]; //LISTA DEL ARCHIVO CARGADO EN ESE MOMENTO

    for(instruccion of lArchivo) {

        instruccion=instruccion.trim()

        listaPrueba.push(instruccion.split(' '))

        listaFile.push(instruccion.split(' '))

    }

    for(l of lArchivo) {

        for(let j = 0; j< l.length; j++) {

            if(l[j] == '') {

                l.splice(j,1);

                j--

            }else{

                l[j] = l[j].trim();

            }

        }

    }

    for(l of listaPrueba) {

      for(let j = 0; j< l.length; j++) {

          if(l[j] == '') {

              l.splice(j,1);

              j--;

          }else {

              l[j] = l[j].trim();

          }

      }

    }

    for(l of listaFile) {

      for(let j = 0; j< l.length; j++) {

          if(l[j] == '') {

              l.splice(j,1);

              j--;

          }else {

              l[j] = l[j].trim();

          }

      }

    }

    let bool = verificarSintaxis(listaPrueba); //bool trae la lista de los errores

    let cf = 0;

    let arrayVariablesIndividual=[];

    let idVariables = Number(listaFile.length)+Number(kernel.value)+1;

    for(l of listaPrueba) {

      if(l[0].toString().toLowerCase() == 'nueva') {

        let valor = [];

          if (l[2] === undefined) {

          alert("La varible no tiene un tipo fijo, por ende no se puede seguir con el proceso, por favor cargue otro archivo");

          location.reload()

          }

        if(l[2].toUpperCase() == 'C') {

          for(let i = 3; i<l.length; i++) {

              valor.push(l[i]);

          }

            valor = valor.toString();

            valor = valor.replaceAll(',', " ");

            let idVariables = Number(listaPrueba.length)+Number(kernel.value)+1;

            arrayVariables.push(new Nueva(idVariables, l[1], l[2], valor, name));

            arrayVariablesIndividual.push(new Nueva(idVariables, l[1], l[2], valor, zeroFill(numId,3)))

            // se agrega en el array de interfaz

            listaPrueba.push([idVariables, arrayVariables[cf].nombre, arrayVariables[cf].valor]); //Aquí se agregan las variables

            cf++;

        } else {

          let idVariables = Number(listaPrueba.length)+Number(kernel.value)+1;

          arrayVariables.push(new Nueva(idVariables, l[1], l[2], l[3], name));

          arrayVariablesIndividual.push(new Nueva(idVariables, l[1], l[2], l[3], zeroFill(numId,3)));

          // se agrega en el array del archivo

          listaPrueba.push([idVariables, arrayVariables[cf].nombre, arrayVariables[cf].valor]);  //Aquí se agregan la variables

          cf++;

        }

      }

    }

    arrayVariablesFile=[];

    arrayEtiquetasFile =[]

//Array de variables en FileCH

    archivoCH.lineas = listaFile;

    sumArchivo = archivoCH.lineas.length;

    for(file of filesCH){

      sumArchivo +=  file.lineas.length;

    }

    for(l of listaFile) {

      if(l[0].toString().toLowerCase() == 'nueva') {

        let valor = [];

        if(l[2].toUpperCase() == 'C') {

            for(let i = 3; i<l.length; i++) {

                valor.push(l[i]);

            }

            valor = valor.toString();

            valor = valor.replaceAll(',', " ");

            arrayVariablesFile.push(new Nueva(idVariables, l[1], l[2], valor, name));

            idVariables++;

            archivoCH.variables = arrayVariablesFile;

            // se agrega en el array de interfaz

            // listaPrueba.push([idVariables, arrayVariables[cf].nombre, arrayVariables[cf].valor]); //Aquí se agregan las variables

            // cf++;

        } else {

            valor = l[3]

            if (valor === undefined){

              valor = "0";

            }else{

              valor = l[3]

            }

            arrayVariablesFile.push(new Nueva(idVariables, l[1], l[2],valor,name));

            archivoCH.variables = arrayVariablesFile;

            idVariables++;

          // se agrega en el array del archivo

          // listaPrueba.push([idVariables, arrayVariables[cf].nombre, arrayVariables[cf].valor]);  //Aquí se agregan la variables

          // cf++;

        }

      }else if(l[0].toString().toLowerCase() == 'etiqueta'){

              let bandera;

              if(l[2] > listaFile.length) {

                  bandera = true;

              } else {

                  bandera = false;

              }

              arrayEtiquetasFile.push(new Etiqueta(l[1], l[2], bandera));

              archivoCH.etiquetas = arrayEtiquetasFile

              listaEtiquetas.push(l);

      }

    }

  archivoCH.fpMemoria = sumArchivo + Number(kernel.value);

  archivoCH.fpvMemoria = sumArchivo + archivoCH.variables.length + Number(kernel.value);

    numVar=arrayVariables.length;

    sum = sum + +sumArchivo + numVar;

    // console.log(sum);

    console.log(`Sum es = ${sum} y kernel es = ${kernel.value}`);

    if(bool.length === 0 && (sum + Number(kernel.value)) <= Number(memoriaInput.value)) {

          for(let i = 0; i<listaPrueba.length; i++) {

            for(let j = 0; j<Number(memoriaInput.value); j++) {

              if(listaPrueba[i][0] == j) {

                listaPrueba[i].splice(0, 1)

              }

            }

          }

          //sin id, con variables, y archivos

      }else {

          if(sum + Number(kernel.value) >  Number(memoriaInput.value)){

            alert(`Error: se excede el espacio de memoria`)

            document.getElementById('instrucciones').innerHTML = 'Error de sintaxis';

            location.reload();

          }

          else{

              for (let err = 0; err < bool.length; err++) {

                alert(bool[err]);

              }

              location.reload()

          }

        }

        filesCH.push(archivoCH)

          for(lines of archivoCH.lineas){

            lines.unshift(initialPosition)

            initialPosition++;

          }

        initialPosition =  archivoCH.fpMemoria + 1

        initialVariables= initialPosition;

        for (let i = 0; i < filesCH.length; i++) {

          for(lineVariable of filesCH[i].variables){

            lineVariable.id = initialVariables;

            initialVariables++;

          }

        }

        console.log(filesCH);

        let lFinal = [];

        let lArchivosCargados =[];

        //llenar el array con todas los archivos cargados dentro del array fileCH

        for (let i = 0; i < filesCH.length; i++) {

          for(instruccion of filesCH[i].lineas){

            lFinal.push(instruccion);

            lArchivosCargados.push(instruccion);

          }

        }

        //llenar el array con todas las variables cargados dentro del array fileCH

        for (let i = 0; i < filesCH.length; i++) {

          for(instruccion of filesCH[i].variables){

            instruccion = Object.values(instruccion)

            instruccion.pop();

            lFinal.push(instruccion);

          }

        }

let caracteres=[]

      //llena de id, la lista a mostrar en el navegador

      let contador = 0;

      let lAcumulador= [0, 'Acumulador']

      let arrayMemoria= [];

      let mostrarOperaciones = [];

      let contenemos;

      let borramos

      let suma = +kernel.value + +lFinal.length ;

      arrayMemoria.push(lAcumulador);

      for(let s=1; s<=Number(memoriaInput.value); s++) {

          if(s<=kernel.value) {

              arrayMemoria.push(`${s} CHSO\_V2021`);

          } else if(s>kernel.value && s<=suma){

            let commentIndex=lFinal[contador].findIndex(comment => comment =='//');

            if (commentIndex !== -1) {

              lFinal[contador].splice(commentIndex,1);

            }

              arrayMemoria.push(lFinal[contador]);

                lFinal[contador].toString().replaceAll(',',' ');

                contador++;

          } else {

              arrayMemoria.push(`${s} - - - - - - `)

          }

      }

      for (string of lArchivosCargados){

        string = string.toString().replaceAll(',',' ');

        mostrarOperaciones.push(string)

      }

      // console.log(arrayMemoria);

      // document.getElementById('memoria').innerHTML = arrayMemoria.join('<br></br>');

      instrucciones = arrayMemoria.slice(+kernel.value+1, +suma+1);

        //Mostrar operaciones en el contenedor sin comas

        let listaVariables = [];

        for (let i = 0; i < filesCH.length; i++) {

          for(variable of filesCH[i].variables){

            listaVariables.push(Object.values(variable));

          }

        }

        // Seguimiento de variables en contenedor MEMORIA

          //ARRAY DE INTERFAZ

        // Agrega en el footer las lineas

          // id

          console.log(filesCH);

        listId.push(zeroFill(numId,3))

        numId++;

          //PRGRAMA

        listPrograma.push(name)

          // INs

        listIns.push(+archivoCH.lineas.length);

        // listIns.push(listaPrueba.length-arrayVariablesIndividual.length)

          //RB

        listRb.push(+archivoCH.ipMemory);

          //RLC

        listRlc.push(+archivoCH.fpMemoria)

          //RLP

        listRlp.push(Number(archivoCH.fpvMemoria))

        // agrega en el interfaz

        idColumn.innerHTML= listId.join('</br>');

        programa.innerHTML= listPrograma.join('</br>');

        ins.innerHTML= listIns.join('</br>');

        rb.innerHTML= listRb.join('</br>');

        rlc.innerHTML= listRlc.join('</br>');

        rlp.innerHTML= listRlp.join('</br>');

      let sinEspacios= [];

      let contar;

      for(m of arrayMemoria) {

        contar = m .toString().replaceAll(',', ' ');

        sinEspacios.push(contar);

      }

      document.getElementById('memoria').innerHTML = sinEspacios.join('<br></br>');

      document.getElementById('instrucciones').innerHTML= mostrarOperaciones.join('<br></br>');

      for(l of listaVariables) {

        let va = document.createElement('span');

        va.append(`${l[0]} ${l[1]}`);

        document.getElementById('variables').appendChild(va);

      }

      for(l of listaEtiquetas) {

        let eti = document.createElement('span');

        eti.append(`${l[0]} ${l[2]}`);

        document.getElementById('etiquetas').appendChild(eti);

      }

      ejecutar.style.display = 'inline-block';

      btnStepbyStep.style.display='inline-block'

      listaArchivos = lFinal;

      })

    }

  }

Esta clase es la que se encarga de abrir los programas .ch, para luego leer línea tras línea de estos agregándolos así en un arreglo de tipo string, es decir por cada posición del arreglo estarán una a una las líneas del programa correspondiente.

**FUNCIÓN PROCESARARCHIVO()**

function procesarArchivo(ch, callback) {

    var reader = new FileReader();

    reader.readAsText(ch);

    reader.onload = function () {

        callback(reader.result);

    }

}

Esta función continua con el debido proceso de cada uno de los archivos con sintaxis del ch-código a la memoria, los cuales se cargarán a la memoria dejando el espacio libre de los que se reservaron para el sistema operativo, garantizando que no se exceda el espacio de memoria.

**FUNCIÓN VERIFICARSINTAXIS()**

function verificarSintaxis(lista) {

Esta función se encarga de cargar el archivo .ch y comenzar a analizar línea por línea de este y verificar su sintaxis y si es correcta lo carga en la tabla de memoria clasifica las variables y las etiquetas y las carga a las tablas correspondientes, si todo sale bien carga el dato del programa a la tabla de procesos.

**FUNCIÓN MOMEMORIA()**

function moMemoria() {

    divMemoria.style.display = 'block';

    cerrar.style.display = 'inline-block';

}

Esta función se encarga de administrar los Sectores de memoria que se ejecutaran durante el funcionamiento del programa, a su vez, permite Mostrar la posición del sector y su contenido al momento de ejecutar el archivo Ingresado.

**FUNCIÓN CERRARDIV()**

function cerrarDiv() {

    divMemoria.style.display = 'none';

    cerrar.style.display = 'none'

}

Esta función utiliza el evento onclick(), es decir, que cuando se ejecuta un click sobre este, procede esconder el contenedor de memoria, y se esconde así mismo.

**FUNCIÓN CORRERARCHIVO()**

Function correrArchivo(acumulador, filesCH){

    btnStepbyStep.style.display='none';

    let listMonitor = [];

    let listPrinter = [];

    let time = 0;

    let namesPrint= ''

    for(let i=0 ; i<filesCH.length; i++){

        let file = filesCH[i];

        // alert(`Ahora corre el programa siguiente que es: ${file.name}`)

        setTimeout(function(){

            namesPrint += `${file.name} `

            for(let instruccion = 0; instruccion<file.lineas.length; instruccion++){

                console.log(file.lineas[instruccion][1]);

                // debugger;

                if (file.lineas[instruccion][1].trim().includes('//')) {

                    continue;

                }

                if(file.lineas[instruccion][1].toLowerCase() =='lea') {//Preguntar al Profe

                    for (let variable = 0; variable < file.variables.length; variable++) {

                        // debugger;

                        if (file.lineas[instruccion][2] == file.variables[variable].nombre) {

                        let newValue = prompt(`Ingrese el VALOR de la variable ${file.variables[variable].nombre}`);

                        file.variables[variable].valor = String(newValue);

                        console.log(file.variables[variable]);

                        }

                    }

                    console.log('lea');

                }else if(file.lineas[instruccion][1].toLowerCase() =='cargue') {

                    // debugger;

                    for (let variable = 0; variable < file.variables.length; variable++) {

                        if (file.lineas[instruccion][2].trim() === file.variables[variable].nombre) {

                        acumulador = String(file.variables[variable].valor);

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='almacene') {

                    for(almacene of file.variables){

                        if(file.lineas[instruccion][2] == almacene.nombre){

                            // console.log(`valor de ${almacene.nombre} = ${almacene.valor}`);

                            almacene.valor = acumulador;

                        }

                    }

                }else if(file.lineas[instruccion][1].trim().toLowerCase() === 'vaya') {

                for(etiquetas of file.etiquetas) {

                    if(etiquetas.nombre== file.lineas[instruccion][2].trim()) {

                        if(etiquetas.sobrepasa == false  && etiquetas.valor < file.lineas[instruccion-1]) {

                            instruccion = Number(etiquetas.valor) - 2;

                        }else{

                            alert(`La etiqueta ${etiquetas.nombre} con un valor de ${etiquetas.valor}, sobrepasa la longitud del archivo que es= ${file.lineas.length}`);

                            location.reload()

                        }

                    }

                }

            }else if(file.lineas[instruccion][1].trim().toLowerCase() =='vayasi') {

                // debugger;

                if(Number(acumulador) > 0) {

                    for(e of file.etiquetas) {

                        if(file.lineas[instruccion][2].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa == false) {

                            console.log('Entró a la recursión');

                            instruccion = Number(e.valor) - 2;

                        }

                    }

                } else if(acumulador < 0) {

                    for(e of file.etiquetas) {

                    if(file.lineas[instruccion][3].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa === false) {

                        console.log('Entró a la recursión');

                        instruccion = Number(e.valor) - 2;

                    }

                    }

                }

                console.log('Me salí x2');

            }else if(file.lineas[instruccion][1].toLowerCase() == 'sume') {

                    for(sume of file.variables){

                        if(file.lineas[instruccion][2] == sume.nombre){

                            acumulador=Number(acumulador)

                            acumulador=Number(acumulador) + Number(sume.valor);

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='reste') {

                    for(reste of file.variables){

                        if(file.lineas[instruccion][2] == reste.nombre){

                            acumulador-= reste.valor;

                        }else if(file.lineas[instruccion][2] == 'acumulador'){

                        acumulador = 0

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='multiplique') {

                    for(multi of file.variables){

                        if(file.lineas[instruccion][2] == multi.nombre){

                            acumulador= acumulador \* multi.valor;

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='divida') {

                    for(div of file.variables){

                        if(file.lineas[instruccion][2] == div.nombre && div.valor != 0){

                            acumulador= acumulador / div.valor;

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='potencia') {

                    for(potencia of file.variables){

                        if(file.lineas[instruccion][2] == potencia.nombre && potencia.valor.isInteger()){

                            acumulador= acumulador \*\* potencia.valor;

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='modulo') {

                    for(mod of file.variables){

                        if(file.lineas[instruccion][2] == mod.nombre){

                            let modulo= acumulador % mod.valor;

                            alert(`El modulo de ${acumulador} % ${mod.valor} = ${modulo}(linea ${mod.id})`);

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='concatene') {

                    for(concatene of file.variables){

                        if(file.lineas[instruccion][2].trim() == concatene.nombre){

                            let cad= acumulador +' '+ concatene.valor;

                            inputAcumulador.type= 'text';

                            acumulador = cad;

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='elimine') {

                    let lol = file.lineas[instruccion][2].trim()

                    acumulador=acumulador.replaceAll(lol,'')

                }else if(file.lineas[instruccion][1].trim() =='Y') {

                    let primerOperando = 0;

                    let segundoOperando = 0;

                    for(v of file.variables) {

                        if(file.lineas[instruccion][2].trim() == v.nombre) {

                            primerOperando = v.valor;

                        }

                        if (file.lineas[instruccion][3].trim() == v.nombre) {

                            segundoOperando = v.valor;

                        }

                        if(file.lineas[instruccion][4].trim() == v.nombre) {

                            if(primerOperando && segundoOperando == 1) {

                                v.valor = 1;

                            } else if (primerOperando && segundoOperando == 0) {

                                v.valor = 0;

                            }

                        }

                    }

                }else if(file.lineas[instruccion][1].trim() =='O') {

                    let primerOperando = 0;

                    let segundoOperando = 0;

                    for(v of file.variables) {

                        if(file.lineas[instruccion][2].trim() == v.nombre) {

                            primerOperando = v.valor;

                            console.log(`El valor de ${v.nombre} = ${v.valor}`);

                        }

                        if (file.lineas[instruccion][3].trim() == v.nombre) {

                            segundoOperando = v.valor;

                            console.log(`El valor de ${v.nombre} = ${v.valor}`)

                        }

                        if(file.lineas[instruccion][4].trim() == v.nombre) {

                            if(primerOperando || segundoOperando == 1) {

                                v.valor = 1;

                                console.log(`El valor de ${v.nombre} = ${v.valor}`)

                            }else if (primerOperando || segundoOperando == 0) {

                                v.valor = 0;

                                console.log(`El valor de ${v.nombre} = ${v.valor}`)

                            }

                        }

                    }

                }else if(file.lineas[instruccion][1].trim() =='NO') {

                    let enNegativo

                    for(v of file.variables){

                    if(file.lineas[instruccion][2].trim()==v.nombre){

                        if(v.tipo == 'L') {

                        if(v.valor == 0) {

                            enNegativo = 1;

                        } else {

                            enNegativo = 0;

                        }

                        }

                    }

                    }

                    console.log(`${file.lineas[instruccion][1]} = ${v.valor}`);

                    for(v of file.variables){

                    if(file.lineas[instruccion][3].trim() == v.nombre) {

                        v.valor = enNegativo;

                    }

                    }

                    console.log(`${file.lineas[instruccion][1]} = ${v.valor}`);

                }else if(file.lineas[instruccion][1].toLowerCase() =='muestre') {

                    if(file.lineas[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                        monitor.innerHTML= `El resultado del(os)

                        programas(${namesPrint})

                        (${file.lineas[instruccion][2]})

                         es: ${acumulador}`;

                    }

                    else{

                        for(muestre of file.variables ){

                            if(file.lineas[instruccion][2].trim() == muestre.nombre){

                                    monitor.innerHTML= `El resultado del(os)

                                    programas(${namesPrint})

                                    (${file.lineas[instruccion][2]}) es igual a --> ${muestre.valor}`;

                            }

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() =='imprima') {

                    if(file.lineas[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                        impresora.innerHTML= `El resultado del(os)

                        programas(${namesPrint})

                        (${file.lineas[instruccion][2]})

                         es: ${acumulador}`;

                    }

                    else{

                        for(muestre of file.variables ){

                            if(file.lineas[instruccion][2].trim() == muestre.nombre){

                                    impresora.innerHTML= `El resultado del(os)

                                    programas(${namesPrint})

                                    (${file.lineas[instruccion][2]}) es igual a --> ${muestre.valor}`;

                            }

                        }

                    }

                }else if(file.lineas[instruccion][1].toLowerCase() == 'extraiga'){

                    longitud= acumulador.length;

                    console.log(`la longitud del acumulador es ${longitud}`);

                    let extraer = [];

                    for(i=0; i<Number(file.lineas[instruccion][2]); i++) {

                        extraer.push(acumulador[i]);

                    }

                    acumulador = extraer.join('');

                }else if(file.lineas[instruccion][1].toLowerCase() =='retorne') {

                    acumulador= 0;

                    showMemory(file.lineas[instruccion][2],file,acumulador);

                    // alert('PROGRAMA TERMINADO')

                    // debugger;

                }else if(file.lineas[instruccion][1].toLowerCase() =='raiz') { /\* toma la raiz cuadrada del código \*/

                    for(raiz of file.variables){

                        if(file.lineas[instruccion][2].trim() == raiz.nombre){

                            let ra= raiz.valor;

                            let resultadoRaiz= Math.sqrt(ra);

                        }

                    }

                    console.log(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

                    alert(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

                }

                console.log(`acumulador> ${acumulador}`);

                showMemory(file.lineas[instruccion][2],file,acumulador);

            }

        },time)

        time+= 5000

    }

}

Esta función es una de las principales, ya que será la encargada de albergar las funciones con las que trabajará el sistema, por ejemplo Multiplicar, Restar, Factorial, etc.

**FUNCIÓN EJECUTARPASOAPASO()**

function ejecutarPasoAPaso(){

  stepByStep(instrucciones,contadorPasoPaso,arrayEtiquetas,arrayVariables);

};

Este método invoca la función STEPBYSTEP creada para realizar el proceso de ejecución paso a paso respectivamente.

**FUNCION STEPBYSTEP()**

function stepByStep(acumulador, filesCH, contadorPasoAPaso){

        let file = filesCH[nextFile];

        console.log(file.name);

        if (file.lineas[contadorPasoAPaso][1].trim().includes('//')) {

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`);

        }

        else if(file.lineas[contadorPasoAPaso][1].toLowerCase() == 'nueva') {

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() == 'etiqueta') {

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }

        else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='lea') {//Preguntar al Profe

            for(let variable = 0; variable < file.variables.length; variable++){

                if (file.lineas[contadorPasoAPaso][2] == file.variables[variable].nombre) {

                    let newValue = prompt(`Ingrese el VALOR de la variable ${file.variables[variable].nombre}`);

                    file.variables[variable].valor = String(newValue);

                }

            }

            console.log('lea');

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='cargue') {

            for (let variable = 0; variable < file.variables.length; variable++) {

                if (file.lineas[contadorPasoAPaso][2].trim() === file.variables[variable].nombre) {

                acumulador = String(file.variables[variable].valor);

                }

            }

            inputAcumulador.value = acumulador;

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='almacene') {

            for(almacene of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == almacene.nombre){

                    almacene.valor = acumulador;

                }

            }

            inputAcumulador.value=acumulador;

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].trim().toLowerCase() === 'vaya') {

            for(etiquetas of file.etiquetas) {

                if(etiquetas.nombre== file.lineas[contadorPasoAPaso][2].trim()) {

                    if(etiquetas.sobrepasa == false  && etiquetas.valor < file.lineas[contadorPasoAPaso - 1]) {

                        contadorPasoAPaso = Number(etiquetas.valor) - 2;

                    }else{

                        alert(`La etiqueta ${etiquetas.nombre} con un valor de ${etiquetas.valor}, sobrepasa la longitud del archivo que es= ${file.lineas.length}`);

                        location.reload()

                    }

                }

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ');

            //confirm(`La instruccion es ${unSpace}`);

        }else if(file.lineas[contadorPasoAPaso][1].trim().toLowerCase() =='vayasi') {

            if(Number(acumulador) > 0) {

                for(e of file.etiquetas) {

                    if(file.lineas[contadorPasoAPaso][2].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa == false) {

                        console.log('Entró a la recursión');

                        contadorPasoAPaso = Number(e.valor) - 2;

                    }

                }

            } else if(acumulador < 0) {

                for(e of file.etiquetas) {

                    if(file.lineas[contadorPasoAPaso][3].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa === false) {

                        console.log('Entró a la recursión');

                        contadorPasoAPaso = Number(e.valor) - 2;

                    }

                }

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ');

            //confirm(`La instruccion es ${unSpace}`);

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='sume') {

            for(sume of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == sume.nombre){

                    acumulador=Number(acumulador)

                    acumulador=Number(acumulador) + Number(sume.valor);

                }

                inputAcumulador.value =acumulador;

            }

            //confirm(`el contador va en: ${contadorPasoAPaso}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='reste') {

            for(reste of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == reste.nombre){

                    acumulador-= reste.valor;

                }else if(file.lineas[contadorPasoAPaso][2] == 'acumulador'){

                    acumulador = 0

                }

                inputAcumulador.value = acumulador;

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='multiplique') {

            for(multi of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == multi.nombre){

                    acumulador= acumulador \* multi.valor;

                }

                inputAcumulador.value = acumulador;

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='divida') {

            for(div of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == div.nombre && div.valor != 0){

                    acumulador= acumulador / div.valor;

                }

                inputAcumulador.value = acumulador;

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='potencia') {

            for(potencia of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == potencia.nombre && potencia.valor.isInteger()){

                    acumulador= acumulador \*\* potencia.valor;

                }

                inputAcumulador.value = acumulador;

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='modulo') {

            for(mod of file.variables){

                if(file.lineas[contadorPasoAPaso][2] == mod.nombre){

                    let modulo= acumulador % mod.valor;

                    alert(`El modulo de ${acumulador} % ${mod.valor} = ${modulo}(linea ${mod.id})`);

                }

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='concatene') {

            for(concatene of file.variables){

                if(file.lineas[contadorPasoAPaso][2].trim() == concatene.nombre){

                    let cad= acumulador +' '+ concatene.valor;

                    inputAcumulador.type= 'text';

                    acumulador = cad;

                    inputAcumulador.value = acumulador;

                }

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='elimine') {

            let lol = file.lineas[contadorPasoAPaso][2].trim()

            acumulador=acumulador.replaceAll(lol,'')

            inputAcumulador.value=acumulador;

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`);

        }else if(file.lineas[contadorPasoAPaso][1].trim() =='Y') {

            let primerOperando = 0;

                    let segundoOperando = 0;

                    for(v of file.variables) {

                        if(file.lineas[contadorPasoAPaso][2].trim() == v.nombre) {

                            primerOperando = v.valor;

                        }

                        if (file.lineas[contadorPasoAPaso][3].trim() == v.nombre) {

                            segundoOperando = v.valor;

                        }

                        if(file.lineas[contadorPasoAPaso][4].trim() == v.nombre) {

                            if(primerOperando && segundoOperando == 1) {

                                v.valor = 1;

                            } else if (primerOperando && segundoOperando == 0) {

                                v.valor = 0;

                            }

                        }

                    }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].trim() =='O') {

            let primerOperando = 0;

                let segundoOperando = 0;

                for(v of file.variables) {

                    if(file.lineas[contadorPasoAPaso][2].trim() == v.nombre) {

                        primerOperando = v.valor;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`);

                    }

                    if (file.lineas[contadorPasoAPaso][3].trim() == v.nombre) {

                        segundoOperando = v.valor;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }

                    if(file.lineas[contadorPasoAPaso][4].trim() == v.nombre) {

                        if(primerOperando || segundoOperando == 1) {

                            v.valor = 1;

                            console.log(`El valor de ${v.nombre} = ${v.valor}`)

                        }else if (primerOperando || segundoOperando == 0) {

                            v.valor = 0;

                            console.log(`El valor de ${v.nombre} = ${v.valor}`)

                        }

                    }

                }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].trim() =='NO') {

            let enNegativo

            for(v of file.variables){

                if(file.lineas[contadorPasoAPaso][2].trim()==v.nombre){

                    if(v.tipo == 'L') {

                    if(v.valor == 0) {

                        enNegativo = 1;

                    } else {

                        enNegativo = 0;

                    }

                    }

                }

            }

            console.log(`${file.lineas[contadorPasoAPaso][1]} = ${v.valor}`);

            for(v of file.variables){

                if(file.lineas[contadorPasoAPaso][3].trim() == v.nombre) {

                    v.valor = enNegativo;

                }

            }

            console.log(`${file.lineas[contadorPasoAPaso][1]} = ${v.valor}`);

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='muestre') {

            if(file.lineas[contadorPasoAPaso][2].toUpperCase().trim() == 'ACUMULADOR'){

                monitor.innerHTML= `El resultado del(os)

                programas(${file.name})

                (${file.lineas[contadorPasoAPaso][2]})

                 es: ${acumulador}`;

            }

            else{

                for(muestre of file.variables ){

                    if(file.lineas[contadorPasoAPaso][2].trim() == muestre.nombre){

                            monitor.innerHTML= `El resultado del(os)

                            programas(${file.name})

                            (${file.lineas[contadorPasoAPaso][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='imprima') {

            if(file.lineas[contadorPasoAPaso][2].toUpperCase().trim() == 'ACUMULADOR'){

                impresora.innerHTML= `El resultado del(os)

                programas(${file.name})

                (${file.lineas[contadorPasoAPaso][2]})

                 es: ${acumulador}`;

            }

            else{

                for(muestre of file.variables ){

                    if(file.lineas[contadorPasoAPaso][2].trim() == muestre.nombre){

                            impresora.innerHTML= `El resultado del(os)

                            programas(${file.name})

                            (${file.lineas[contadorPasoAPaso][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() == 'extraiga'){

            longitud= acumulador.length;

                console.log(`la longitud del acumulador es ${longitud}`);

                let extraer = [];

                for(i=0; i<Number(file.lineas[contadorPasoAPaso][2]); i++) {

                    extraer.push(acumulador[i]);

                }

                acumulador = extraer.join('');

            inputAcumulador.value = acumulador;

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase().trim() =='retorne') {

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

            if(Number(nextFile + 1) === Number(filesCH.length)){

                btnStepbyStep.style.display='none';

            }

            nextFile++;

            contadorPasoAPaso = - 1;

        }else if(file.lineas[contadorPasoAPaso][1].toLowerCase() =='raiz') { /\* toma la raiz cuadrada del código \*/

            for(raiz of file.variables){

                if(file.lineas[contadorPasoAPaso][2].trim() == raiz.nombre){

                    let ra= raiz.valor;

                    let resultadoRaiz= Math.sqrt(ra);

                }

            }

            console.log(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

            alert(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

            let unSpace= file.lineas[contadorPasoAPaso].toString().replaceAll(',',' ')

            //confirm(`La instruccion es ${unSpace}`)

        }

        if (contadorPasoAPaso !== - 1){

            showMemory(file.lineas[contadorPasoAPaso][2],file, acumulador);

        }

        acumStepbyStep = acumulador;

        contPasoApaso = contadorPasoAPaso;

        contPasoApaso++;

}

**CLASS NUEVA.JS**

Class Nueva {

    constructor(id, nombre, tipo, valor) {

        this.\_id = id;

        this.\_nombre = nombre;

        this.\_tipo = tipo;

        this.\_valor= valor;

    }

    get id() {

        return this.\_id;

    }

    set id (id) {

        this.\_id = id;

    }

    get nombre() {

        return this.\_nombre;

    }

    set nombre (nombre) {

        this.\_nombre = nombre;

    }

    get tipo() {

        return this.\_tipo;

    }

    set tipo (tipo) {

        this.\_tipo = tipo;

    }

    get valor() {

        return this.\_valor;

    }

    set valor (valor) {

        this.\_valor = valor;

    }

}

En esta clase se crean las variables a utilizar en el objeto.

**CLASS ETIQUETA.JS**

class Etiqueta {

    constructor(id, nombre, valor, sobrepasa) {

        this.\_id = id;

        this.\_nombre = nombre;

        this.\_sobrepasa = sobrepasa;

        this.\_valor= valor;

    }

    get id() {

        return this.\_id;

    }

    set id (id) {

        this.\_id = id;

    }

    get nombre() {

        return this.\_nombre;

    }

    set nombre (nombre) {

        this.\_nombre = nombre;

    }

    get sobrepasa() {

        return this.\_sobrepasa;

    }

    set sobrepasa (sobrepasa) {

        this.\_sobrepasa = sobrepasa;

    }

    get valor() {

        return this.\_valor;

    }

    set valor (valor) {

        this.\_valor = valor;

    }

}

En esta clase se crean las varianbles a utilizar en la funcion VAYASI para ejecutarlo satisfactoriamente.

function zeroFill( number, width )

{

  width -= number.toString().length;

  if ( width > 0 )

  {

    return new Array( width + (/./.test( number ) ? 2 : 1) ).join( '0' ) + number;

  }

  return number + ""; // always return a string

}

function ejecutarPrograma(){

  console.log('entra al ejecutar porgrama');

  correrArchivo(instrucciones, 0,instrucciones.length, arrayEtiquetas, arrayVariables,0);

  ejecutar.style.display= 'none'

}

Esta funcion es llamada por la función LEERARCHIVO(), su objetivo es agregar ceros a la izquierda de un numero.

**FUNCION PLANIFICACIONDEALGORITMOS()**

function ordenarAlgoritmos  (filesCH, algorithmToUse, quantum){

    console.log(quantum);

    console.log(filesCH);

    console.log(algorithmToUse);

    if (algorithmToUse === 'fcfs'){ //orden de llegada

        filesCH = filesCH.sort((a, b) => Number(a.id) - Number(b.id));

        console.log(`entra al FCFS`);

    }else if (algorithmToUse === 'sjf'){ //dependiendo el n[umero de linea de menor a mayor

        filesCH = filesCH.sort((a, b) => Number(a.id) - Number(b.id));

        //   contId2 = +filesCH[filesCH.length - 1].fpvMemory + 1;

      filesCH = filesCH.sort(

        (a, b) => Number(a.lineas.length) - Number(b.lineas.length)

      );

    }else if (algorithmToUse === 'prioridad') { // dependiendo la prioridad si es mayor es mas importante

        filesCH = filesCH.sort((a, b) => Number(a.id) - Number(b.id));

        // contId2 = +filesCH[filesCH.length - 1].fpvMemoria + 1;

        filesCH = filesCH.sort(

          (a, b) => Number(b.priority) - Number(a.priority)

        );

    }else if (algorithmToUse === 'RR') {

        for (let file of filesCH) {

          file.endingRr = quantum;

        }

        filesCH.sort((a, b) => Number(a.id) - Number(b.id));

    }else if (algorithmToUse === 'RRP') {

        filesCH = filesCH.sort((a, b) => Number(a.id) - Number(b.id));

        filesCH = filesCH.sort(

          (a, b) => Number(b.priority) - Number(a.priority)

        );

        for (let file of filesCH) {

          file.endingRr = quantum;

        }

    }else if (algorithmToUse === 'srtn') {

        filesCH.sort((a, b) => Number(a.id) - Number(b.id));

        // contId2 = +filesArrayFake[filesArrayFake.length - 1].fpvMemory + 1;

      }

}

En esta función se organizan los procesos a cargar en cada uno de los algoritmos de planificación teniendo en cuenta las condiciones que utiliza cada uno.

**FUNCION RUNSRTN()**

filesStateList= [];

function runSrtn(filesCH) {

    let acumulador= '0';

    for (let file = 0; file < filesCH.length; file++) {

      if (filesStateList[file] === undefined) {

        filesStateList.push(filesCH[file]);

      }

      if (filesStateList.length === filesCH.length) {

        filesStateList = filesStateList.sort(

          (a, b) => Number(a.cpu) - Number(b.cpu)

        );

      }

      if (filesCH.length > 1 && file < filesCH.length - 1) {

        filesStateList = filesStateList.sort(

          (a, b) => Number(a.cpu) - Number(b.cpu)

        );

        file = 0;

        filesStateList[file].endingRr = filesCH[file + 1].ti - 1;

      } else {

        filesStateList[0].endingRr = filesStateList[0].lineas.length - 1;

        filesStateList[0].cpu = 1000;

        file = 0;

      }

      acumulador = filesStateList[0].acumulator;

      for(let instruccion = filesStateList[file].initialRr; instruccion < filesStateList[file].endingRr; instruccion++){

        console.log(filesStateList[file].lineas[instruccion][1]);

        console.log(filesStateList[file].name, filesStateList[file].lineas[instruccion]);

        if (filesStateList[file].lineas[instruccion][1].trim().includes('//')) {

            continue;

        }

        if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='lea') {

            for (let variable = 0; variable < filesStateList[file].variables.length; variable++) {

                // debugger;

                if (filesStateList[file].lineas[instruccion][2] == filesStateList[file].variables[variable].nombre) {

                let newValue = prompt(`Ingrese el VALOR de la variable ${filesStateList[file].variables[variable].nombre}`);

                filesStateList[file].variables[variable].valor = String(newValue);

                console.log(filesStateList[file].variables[variable]);

                }

            }

            console.log('lea');

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='cargue') {

            // debugger;

            for (let variable = 0; variable < filesStateList[file].variables.length; variable++) {

                if (filesStateList[file].lineas[instruccion][2].trim() === filesStateList[file].variables[variable].nombre) {

                acumulador = String(filesStateList[file].variables[variable].valor);

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='almacene') {

            for(almacene of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == almacene.nombre){

                    // console.log(`valor de ${almacene.nombre} = ${almacene.valor}`);

                    almacene.valor = acumulador;

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].trim().toLowerCase() === 'vaya') {

        for(etiquetas of filesStateList[file].etiquetas) {

            if(etiquetas.nombre== filesStateList[file].lineas[instruccion][2].trim()) {

                if(etiquetas.sobrepasa == false  && etiquetas.valor < filesStateList[file].lineas[instruccion-1]) {

                    instruccion = Number(etiquetas.valor) - 2;

                }else{

                    alert(`La etiqueta ${etiquetas.nombre} con un valor de ${etiquetas.valor}, sobrepasa la longitud del archivo que es= ${filesStateList[file].lineas.length}`);

                    location.reload()

                }

            }

        }

    }else if(filesStateList[file].lineas[instruccion][1].trim().toLowerCase() =='vayasi') {

        // debugger;

        if(Number(acumulador) > 0) {

            for(e of filesStateList[file].etiquetas) {

                if(filesStateList[file].lineas[instruccion][2].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa == false) {

                    console.log('Entró a la recursión');

                    instruccion = Number(e.valor) - 2;

                }

            }

        } else if(acumulador < 0) {

            for(e of filesStateList[file].etiquetas) {

            if(filesStateList[file].lineas[instruccion][3].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa === false) {

                console.log('Entró a la recursión');

                instruccion = Number(e.valor) - 2;

            }

            }

        }

        console.log('Me salí x2');

    }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() == 'sume') {

            for(sume of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == sume.nombre){

                    acumulador=Number(acumulador)

                    acumulador=Number(acumulador) + Number(sume.valor);

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='reste') {

            for(reste of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == reste.nombre){

                    acumulador-= reste.valor;

                }else if(filesStateList[file].lineas[instruccion][2] == 'acumulador'){

                acumulador = 0

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='multiplique') {

            for(multi of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == multi.nombre){

                    acumulador= acumulador \* multi.valor;

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='divida') {

            for(div of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == div.nombre && div.valor != 0){

                    acumulador= acumulador / div.valor;

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='potencia') {

            for(potencia of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == potencia.nombre && potencia.valor.isInteger()){

                    acumulador= acumulador \*\* potencia.valor;

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='modulo') {

            for(mod of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2] == mod.nombre){

                    let modulo= acumulador % mod.valor;

                    alert(`El modulo de ${acumulador} % ${mod.valor} = ${modulo}(linea ${mod.id})`);

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='concatene') {

            for(concatene of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2].trim() == concatene.nombre){

                    let cad= acumulador +' '+ concatene.valor;

                    inputAcumulador.type= 'text';

                    acumulador = cad;

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='elimine') {

            let lol = filesStateList[file].lineas[instruccion][2].trim()

            acumulador=acumulador.replaceAll(lol,'')

        }else if(filesStateList[file].lineas[instruccion][1].trim() =='Y') {

            let primerOperando = 0;

            let segundoOperando = 0;

            for(v of filesStateList[file].variables) {

                if(filesStateList[file].lineas[instruccion][2].trim() == v.nombre) {

                    primerOperando = v.valor;

                }

                if (filesStateList[file].lineas[instruccion][3].trim() == v.nombre) {

                    segundoOperando = v.valor;

                }

                if(filesStateList[file].lineas[instruccion][4].trim() == v.nombre) {

                    if(primerOperando && segundoOperando == 1) {

                        v.valor = 1;

                    } else if (primerOperando && segundoOperando == 0) {

                        v.valor = 0;

                    }

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].trim() =='O') {

            let primerOperando = 0;

            let segundoOperando = 0;

            for(v of filesStateList[file].variables) {

                if(filesStateList[file].lineas[instruccion][2].trim() == v.nombre) {

                    primerOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`);

                }

                if (filesStateList[file].lineas[instruccion][3].trim() == v.nombre) {

                    segundoOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`)

                }

                if(filesStateList[file].lineas[instruccion][4].trim() == v.nombre) {

                    if(primerOperando || segundoOperando == 1) {

                        v.valor = 1;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }else if (primerOperando || segundoOperando == 0) {

                        v.valor = 0;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].trim() =='NO') {

            let enNegativo

            for(v of filesStateList[file].variables){

            if(filesStateList[file].lineas[instruccion][2].trim()==v.nombre){

                if(v.tipo == 'L') {

                if(v.valor == 0) {

                    enNegativo = 1;

                } else {

                    enNegativo = 0;

                }

                }

            }

            }

            console.log(`${filesStateList[file].lineas[instruccion][1]} = ${v.valor}`);

            for(v of filesStateList[file].variables){

            if(filesStateList[file].lineas[instruccion][3].trim() == v.nombre) {

                v.valor = enNegativo;

            }

            }

            console.log(`${filesStateList[file].lineas[instruccion][1]} = ${v.valor}`);

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='muestre') {

            if(filesStateList[file].lineas[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                monitor.innerHTML= `El resultado del(os)

                programas(${filesStateList[file].name})

                (${filesStateList[file].lineas[instruccion][2]})

                 es: ${acumulador}`;

            }

            else{

                for(muestre of filesStateList[file].variables ){

                    if(filesStateList[file].lineas[instruccion][2].trim() == muestre.nombre){

                            monitor.innerHTML= `El resultado del(os)

                            programas(${filesStateList[file].name})

                            (${filesStateList[file].lineas[instruccion][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='imprima') {

            if(filesStateList[file].lineas[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                impresora.innerHTML= `El resultado del(os)

                programas(${filesStateList[file].name})

                (${filesStateList[file].lineas[instruccion][2]})

                 es: ${acumulador}`;

            }

            else{

                for(muestre of filesStateList[file].variables ){

                    if(filesStateList[file].lineas[instruccion][2].trim() == muestre.nombre){

                            impresora.innerHTML= `El resultado del(os)

                            programas(${filesStateList[file].name})

                            (${filesStateList[file].lineas[instruccion][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() == 'extraiga'){

            longitud= acumulador.length;

            console.log(`la longitud del acumulador es ${longitud}`);

            let extraer = [];

            for(i=0; i<Number(filesStateList[file].lineas[instruccion][2]); i++) {

                extraer.push(acumulador[i]);

            }

            acumulador = extraer.join('');

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='retorne') {

            acumulador= 0;

            showMemory(filesStateList[file].lineas[instruccion][2],file,acumulador);

            // alert('PROGRAMA TERMINADO')

            // debugger;

        }else if(filesStateList[file].lineas[instruccion][1].toLowerCase() =='raiz') { /\* toma la raiz cuadrada del código \*/

            for(raiz of filesStateList[file].variables){

                if(filesStateList[file].lineas[instruccion][2].trim() == raiz.nombre){

                    let ra= raiz.valor;

                    let resultadoRaiz= Math.sqrt(ra);

                }

            }

            console.log(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

            alert(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

        }

        // console.log(`acumulador> ${acumulador}`);

        showMemory(filesStateList[file].lineas[instruccion][2],filesStateList[file],acumulador);

      }

      filesStateList[file].initialRr = filesStateList[file].endingRr;

      filesStateList[file].cpu -= filesStateList[file].endingRr - filesStateList[file].ti;

      filesStateList[file].acumulator = acumulador;

      filesStateList[file] = filesStateList[file];

      for (let miniFile of filesStateList) {

        if (miniFile.cpu > 500) {

          file = filesCH.length;

        } else {

          if (filesStateList.length === filesCH.length) {

            file = filesStateList.length - 2;

          } else {

            file = filesStateList.length - 1;

          }

          break;

        }

      }

      // debugger;

    }

    for (let miniStateList = 0; miniStateList < filesStateList.length; miniStateList++) {

      filesCH[miniStateList] = filesStateList[miniStateList];

    }

    return [filesCH];

}

Este algoritmo siempre ejecuta primero aquellos procesos a los que les queda menos tiempo

 para terminar Este algoritmo también es conocido como ‘optimo’, pues con el se obtienen

los mejores resultados

Esta función está diseñada para evaluar y llevar a cabo las condiciones del algoritmo SRTN de planificacion. Inicialmente se inicia un FileStateList [] que se encarga de guardar los arrays, los ciclos for evaluan si el FileStateList[] esta vacío y posiciona el primer proceso. Seguido a esto se actualizan las propiedades del archivo. Despues de que sube el primer archivo y corre al menos una vez todos los file y si aun no es el ultimo se evalua y posiciona, a medida de que avanza posiciona según las rafagas que tenga cada proceso.

Para llevar a cabo la ejecucion de este algoritmo de planificacion se debe tener en cuenta el algoritmo SJF,

**FUNCION ROUNDROBIN()**

filesStateList = [];

quantumCounter = 0;

function roundRobin (filesCH, quantum){

    let acumulador

    for (let file = 0; file < filesCH.length; file++) {

      if (filesStateList.length < filesCH.length) {

        acumulador = '0';

      } else {

        acumulador = filesStateList[file][1];

      }

      for(let instruccion = filesCH[file].initialRr; instruccion < filesCH[file].endingRr; instruccion++){

        console.log(filesCH[file].lineas[instruccion][1]);

        console.log(filesCH[file].name, filesCH[file].lineas[instruccion]);

        if (filesCH[file].lineas[instruccion][1].trim().includes('//')) {

            continue;

        }

        if(filesCH[file].lineas[instruccion][1].toLowerCase() =='lea') {

            for (let variable = 0; variable < filesCH[file].variables.length; variable++) {

                // debugger;

                if (filesCH[file].lineas[instruccion][2] == filesCH[file].variables[variable].nombre) {

                let newValue = prompt(`Ingrese el VALOR de la variable ${filesCH[file].variables[variable].nombre}`);

                filesCH[file].variables[variable].valor = String(newValue);

                }

            }

            console.log('lea');

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='cargue') {

            // debugger;

            for (let variable = 0; variable < filesCH[file].variables.length; variable++) {

                if (filesCH[file].lineas[instruccion][2].trim() === filesCH[file].variables[variable].nombre) {

                acumulador = String(filesCH[file].variables[variable].valor);

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='almacene') {

            for(almacene of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == almacene.nombre){

                    // console.log(`valor de ${almacene.nombre} = ${almacene.valor}`);

                    almacene.valor = acumulador;

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].trim().toLowerCase() === 'vaya') {

        for(etiquetas of filesCH[file].etiquetas) {

            if(etiquetas.nombre== filesCH[file].lineas[instruccion][2].trim()) {

                if(etiquetas.sobrepasa == false  && etiquetas.valor < filesCH[file].lineas[instruccion-1]) {

                    instruccion = Number(etiquetas.valor) - 2;

                }else{

                    alert(`La etiqueta ${etiquetas.nombre} con un valor de ${etiquetas.valor}, sobrepasa la longitud del archivo que es= ${filesCH[file].lineas.length}`);

                    location.reload()

                }

            }

        }

    }else if(filesCH[file].lineas[instruccion][1].trim().toLowerCase() =='vayasi') {

        // debugger;

        if(Number(acumulador) > 0) {

            for(e of filesCH[file].etiquetas) {

                if(filesCH[file].lineas[instruccion][2].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa == false) {

                    console.log('Entró a la recursión');

                    instruccion = Number(e.valor) - 2;

                }

            }

        } else if(acumulador < 0) {

            for(e of filesCH[file].etiquetas) {

            if(filesCH[file].lineas[instruccion][3].trim().toLowerCase() == e.nombre.trim().toLowerCase() && e.sobrepasa === false) {

                console.log('Entró a la recursión');

                instruccion = Number(e.valor) - 2;

            }

            }

        }

        console.log('Me salí x2');

    }else if(filesCH[file].lineas[instruccion][1].toLowerCase() == 'sume') {

            for(sume of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == sume.nombre){

                    acumulador=Number(acumulador)

                    acumulador=Number(acumulador) + Number(sume.valor);

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='reste') {

            for(reste of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == reste.nombre){

                    acumulador-= reste.valor;

                }else if(filesCH[file].lineas[instruccion][2] == 'acumulador'){

                acumulador = 0

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='multiplique') {

            for(multi of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == multi.nombre){

                    acumulador= acumulador \* multi.valor;

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='divida') {

            for(div of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == div.nombre && div.valor != 0){

                    acumulador= acumulador / div.valor;

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='potencia') {

            for(potencia of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == potencia.nombre && potencia.valor.isInteger()){

                    acumulador= acumulador \*\* potencia.valor;

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='modulo') {

            for(mod of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2] == mod.nombre){

                    let modulo= acumulador % mod.valor;

                    alert(`El modulo de ${acumulador} % ${mod.valor} = ${modulo}(linea ${mod.id})`);

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='concatene') {

            for(concatene of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2].trim() == concatene.nombre){

                    let cad= acumulador +' '+ concatene.valor;

                    inputAcumulador.type= 'text';

                    acumulador = cad;

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='elimine') {

            let lol = filesCH[file].lineas[instruccion][2].trim()

            acumulador=acumulador.replaceAll(lol,'')

        }else if(filesCH[file].lineas[instruccion][1].trim() =='Y') {

            let primerOperando = 0;

            let segundoOperando = 0;

            for(v of filesCH[file].variables) {

                if(filesCH[file].lineas[instruccion][2].trim() == v.nombre) {

                    primerOperando = v.valor;

                }

                if (filesCH[file].lineas[instruccion][3].trim() == v.nombre) {

                    segundoOperando = v.valor;

                }

                if(filesCH[file].lineas[instruccion][4].trim() == v.nombre) {

                    if(primerOperando && segundoOperando == 1) {

                        v.valor = 1;

                    } else if (primerOperando && segundoOperando == 0) {

                        v.valor = 0;

                    }

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].trim() =='O') {

            let primerOperando = 0;

            let segundoOperando = 0;

            for(v of filesCH[file].variables) {

                if(filesCH[file].lineas[instruccion][2].trim() == v.nombre) {

                    primerOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`);

                }

                if (filesCH[file].lineas[instruccion][3].trim() == v.nombre) {

                    segundoOperando = v.valor;

                    console.log(`El valor de ${v.nombre} = ${v.valor}`)

                }

                if(filesCH[file].lineas[instruccion][4].trim() == v.nombre) {

                    if(primerOperando || segundoOperando == 1) {

                        v.valor = 1;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }else if (primerOperando || segundoOperando == 0) {

                        v.valor = 0;

                        console.log(`El valor de ${v.nombre} = ${v.valor}`)

                    }

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].trim() =='NO') {

            let enNegativo

            for(v of filesCH[file].variables){

            if(filesCH[file].lineas[instruccion][2].trim()==v.nombre){

                if(v.tipo == 'L') {

                if(v.valor == 0) {

                    enNegativo = 1;

                } else {

                    enNegativo = 0;

                }

                }

            }

            }

            console.log(`${filesCH[file].lineas[instruccion][1]} = ${v.valor}`);

            for(v of filesCH[file].variables){

            if(filesCH[file].lineas[instruccion][3].trim() == v.nombre) {

                v.valor = enNegativo;

            }

            }

            console.log(`${filesCH[file].lineas[instruccion][1]} = ${v.valor}`);

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='muestre') {

            if(filesCH[file].lineas[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                monitor.innerHTML= `El resultado del(os)

                programas(${filesCH[file].name})

                (${filesCH[file].lineas[instruccion][2]})

                 es: ${acumulador}`;

            }

            else{

                for(muestre of filesCH[file].variables ){

                    if(filesCH[file].lineas[instruccion][2].trim() == muestre.nombre){

                            monitor.innerHTML= `El resultado del(os)

                            programas(${filesCH[file].name})

                            (${filesCH[file].lineas[instruccion][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='imprima') {

            if(filesCH[file].lineas[instruccion][2].toUpperCase().trim() == 'ACUMULADOR'){

                impresora.innerHTML= `El resultado del(os)

                programas(${filesCH[file].name})

                (${filesCH[file].lineas[instruccion][2]})

                 es: ${acumulador}`;

            }

            else{

                for(muestre of filesCH[file].variables ){

                    if(filesCH[file].lineas[instruccion][2].trim() == muestre.nombre){

                            impresora.innerHTML= `El resultado del(os)

                            programas(${filesCH[file].name})

                            (${filesCH[file].lineas[instruccion][2]}) es igual a --> ${muestre.valor}`;

                    }

                }

            }

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() == 'extraiga'){

            longitud= acumulador.length;

            console.log(`la longitud del acumulador es ${longitud}`);

            let extraer = [];

            for(i=0; i<Number(filesCH[file].lineas[instruccion][2]); i++) {

                extraer.push(acumulador[i]);

            }

            acumulador = extraer.join('');

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='retorne') {

            acumulador= 0;

            showMemory(filesCH[file].lineas[instruccion][2],file,acumulador);

            // alert('PROGRAMA TERMINADO')

            // debugger;

        }else if(filesCH[file].lineas[instruccion][1].toLowerCase() =='raiz') { /\* toma la raiz cuadrada del código \*/

            for(raiz of filesCH[file].variables){

                if(filesCH[file].lineas[instruccion][2].trim() == raiz.nombre){

                    let ra= raiz.valor;

                    let resultadoRaiz= Math.sqrt(ra);

                }

            }

            console.log(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

            alert(`La Raiz Cuadrada del Número${raiz.valor} = ${resultadoRaiz}`);

        }

        // console.log(`acumulador> ${acumulador}`);

        // debugger;

        showMemory(filesCH[file].lineas[instruccion][2],filesCH[file],acumulador);

      }

      filesCH[file].initialRr += quantum;

      filesCH[file].endingRr += quantum;

      if (filesCH[file].endingRr > filesCH[file].lineas.length) {

        filesCH[file].endingRr = filesCH[file].lineas.length;

        if (quantumCounter !== quantum) {

          quantumCounter = quantum - 1;

        }

      }

      if (filesStateList.length < filesCH.length) {

        filesStateList.push([filesCH[file], acumulador]);

      }

      filesStateList[file] = [filesCH[file], acumulador];

      if (file === filesCH.length - 1 && quantumCounter < quantum) {

        file = -1;

        quantumCounter += 1;

      }

    }

    return [filesCH];

}

El algoritmo Round Robin Utiliza una organización en cola circular: Los procesos se ejecutan en cola y cuando acaba el último se sigue con el primero. A cada proceso se le asigna un tiempo de uso de CPU denominado quantum.

En el algoritmo de planificación round robín entra todo el array ya que se debe controlar cada proceso. Con el primer For recorren los procesos al menos una vez y se almacenan en instancias las propiedades de cada uno. También se debe tener un acumulador en el que se pueda recuperar los valores para utilizar más adelante, se asigna por defecto un Quantum de 5 y al iniciar todos comienzan en serio es decir de cero a 5. Después de recorrer todos los que archivos te modifican modifican las variables que almacenan los valores en los que quedaron los procesos.

En las lineas finales se asegura que al terminar de recorrer los procesos, algunos no han acabado al llegar al ultimo se vuelva a comenzar.