Universidad Modelo

UNIVERSIDAD MODELO

Escuela de Ingeniería
Ingeniería en Desarrollo de Tecnología y Software

Materia: Algoritmos
Docente: M.C. Edson G. Estrada López

Reporte de proyecto #1
Métodos de ordenamiento: Algoritmo Infijo – Postfijo

Integrantes:
Hanna Lizarraga Ceballos
Manuel Jesús Canul Witzil

Fecha de elaboración del reporte: 15/05/2019

Objetivo

Objetivos

Objetivo general
Desarrollar nuestras habilidades de programación en JavaScript mediante la implementación de un programa que lea ecuaciones infijas, las pueda convertir a postfijas y resolverlas.

Objetivos particulares
1. Practicar el concepto de pilas para poder implementarlo en este y proyectos mas adelante.
2. Aprender un nuevo tema en programación denominado pilas, para poder tener bases que nos ayudaran en el siguiente semestre.

Documentación

Definición general del proyecto

En equipos de dos personas, implementar un algoritmo que lea ecuaciones infijas, las pueda convertir a postfijas y resolverlas.
El programa debe de poder resolver la ecuación colocada.

Especificación de requerimientos

- El programa debe ser realizado utilizando el lenguaje de programación JavaScript.
- El programa debe evaluar el RPN.
- El programa debe permitir variables.
- El programa debe permitir numeros de multiples cifras.
- El programa debe permitir y evaluar los operadores.
- El programa debe evaluar los parentesis.
- El programa debe manejar errores.

Alcance del proyecto

El programa realizado es capaz de realizar todos los puntos anteriores mencionados en “Especificacion de requerimentos” a excepcion del punto de permitir variables.

Listado de herramientas utilizadas

- Visual Studio Code
- Node.js

Descripción de la arquitectura del sistema

Código:

```
"use string";
// el postfijo no se forma bien con condicionales (solo con operaciones aritmeticas)
no me calcula bien el resultado
var espacios = {"a" : 0};
function main(){

  var rs = require('readline');

  const resp = rs.createInterface({
      input: process.stdin,
      output: process.stdout
    });

    resp.question('Introduzca lo que vamos a calcular: ', (VALO) => {

      console.log("Validando parentesís...");
      let expr = "2 * 3 *(3/3)";
      let {error, index} = verify(VALO);
      if (error === false) {

        console.log('¡Parentesis correctos!');
        console.log();
        console.log("Evaluando Operandos...");

          console.log("¡Operandos correctos!")
          console.log();
          console.log("Separando ecuacion...");
          console.log(tokenize(VALO));
          console.log("¡Ecuacion separada!")
          console.log("Evaluando signos no permitidos...")
          let {consigers,  erers} = validarErrSig(VALO);

          if(consigers === true){
              //console.log("¡Se detectó un error en la expresión!");
              console.log("¡Se hallaron caracteres invalidos!");
              console.log(erers);
          }else{
            console.log("¡No se detecto signo no permitido!")
            console.log();
            console.log("Pasando a Postfijo...");
```

```
            console.log(infixToPostfix(tokenize(VALO)));
            console.log();
            console.log("Evaluando y resolviendo RPN...");
            console.log(RPN(infixToPostfix(tokenize(VALO))));
            console.log();
            console.log("FINALIZADO");
        }
      } else {
        console.log();
        console.log('¡Se detectó un error en los parentesis, verifique su
expresion!');
      }
      resp.close();
    });

}

function verify(expr) {
let stack = [];
const allSymbols = ['{', '[', '(', ')', ']', '}'];
const openSymbols = ['{', '[', '('];
const checkSymbols = { '{': '}', '[': ']', '(': ')' };

for (let i = 0; i < expr.length; ++i) {
let token = expr[i];
if (allSymbols.findIndex(e => e === token) >= 0) {
    if (openSymbols.findIndex(e => e === token) >= 0) {
        stack.push(token);
    } else {
        if (stack.length === 0) {
            return { error: true, index: i };
        }

        let poppedToken = stack.pop();
        if (checkSymbols[poppedToken] !== token) {
            return { error: true, index: i };
        }
    }
}
}

if (stack.length > 0) {
return { error: true, index: expr.length };
}

return { error: false, index: -1 };
}
```

```
//Convierte el arreglo original a postfijo
function infixToPostfix(infix){
const presedences = ["-", "+", "*", "/", "^", ">", "<",  ">=", "<=", "&&", "||"];

var opsStack = [],
  postfix = [];

for(let token of infix){
  // Step 1

  // if("number" === typeof token){
  //if(presedences.includes(token)){
  if(token != "+" && token != "*" && token != "-" && token != "/" && token != "(" &&
  token != ")" && token != "^" && token != ">" && token != "<" && token != ">=" &&
  token != "<=" && token != "&&" && token != "||"){
      postfix.push(token); continue;
  }
  let topOfStack = opsStack[opsStack.length - 1];

  // Step 2
  // Si abre parentesis agregar a arreglo provisional
  if(!opsStack.length || topOfStack == "("){
      opsStack.push(token); continue;
  }
  // Step 3
  if(token == "("){
      opsStack.push(token); continue;
  }
  // Step 4
  if(token == ")"){
    //si cierra parentesis obtiene dato de arreglo provisional y lo mete a arreglo
postfix
      while(opsStack.length){
          let op = opsStack.pop();
          if(op == "(") break;
          postfix.push(op);
      }
      continue;
  }
  // Step 5
  let prevPresedence = presedences.indexOf(topOfStack),
      currPresedence = presedences.indexOf(token);
  while(currPresedence < prevPresedence){
      let op = opsStack.pop();
      postfix.push(op);
      prevPresedence = presedences.indexOf(opsStack[opsStack.length - 1]);
  }
  opsStack.push(token);
```

```
}
// Step 6
while(opsStack.length){
  let op = opsStack.pop();
  if(op == "(") break;
  postfix.push(op);
}

return postfix;
}

// Método que divide la cadena principal en un arreglo con todos los componentes a
evaluar
function tokenize(exp){
  exp = exp.replace(/\s/g, "");
  //reemplazar || por OR para evaluar expresion regular
  exp = exp.replace("||", "OR");

  //Expresion regular que evalua y divide la cadena en un arreglo con todas las partes
a covnertir a postfijo
  var outs = exp.match(/\d+|OR|&&|<=|>=|[^]|[\+-\/x()<>=*]/g);

  // For que sustituye "OR" por "||", se pone asi ya que en la expresion regular no se
pudo reconocer "||".
  for(var c = 0; c < outs.length; c++) {
      if(outs[c] == "OR") {
          outs[c] = "||";
      }
  }
  return outs;
}

function log(obj){
document.querySelector("pre").textContent += JSON.stringify(obj) + "\n";
}



function addition(a, b) {
  return a + b;
}

function multiply(a, b) {
  return a * b;
}

function subtract(a, b) {
  return a - b;
```

```
}

function divide(a, b) {
  return a / b;
}

function exponent(a,b){
  return Math.pow(b, a)
}
//Este metodo resuelve el problema teniendolo ya en postfijo
function RPN(input) {
  let stack = [];
  for (let i=0; i < input.length; i++) {
      element = parseFloat(input[i]);
      if (!Number.isNaN(element)) {
          stack.push(element);
      } else {
          if (input[i] === '+') {
              let b = stack.pop();
              let a = stack.pop();
              stack.push(addition(a, b));
          }
          if (input[i] === '*') {
              let b = stack.pop();
              let a = stack.pop();
              stack.push(multiply(a, b));
          }
          if (input[i] === '/') {
              let b = stack.pop();
              let a = stack.pop();
              stack.push(divide(a, b));
          }
          if (input[i] === '-') {
              let b = stack.pop();
              let a = stack.pop();
              stack.push(subtract(a, b));
          }

          if (input[i] === '^') {
              let b = stack.pop();
              let a = stack.pop();
              stack.push(exponent(b, a));
          }

      }
  }

  return stack;
```

```
}

//Valida si los caracteres son correctos, caracteres ajenos van a ser invalidos
function validarErrSig(ecu){ // 2 + 2
let erss = [];
let ers = 0;
let signos = ['{','[', '(', ')', ']', '}','+', '-', '*', '/', ' ','=',
'^','|','||','&','&&', '<', '>', '<>', '<=',
'>=','1','2','3','4','5','6','7','8','9','0'];

for (let i = 0; i < ecu.length; i++) {
if(signos.includes(ecu[i])){

}else{
  erss.push(ecu[i]);
  ers++;
}
}

if(ers > 0){
return {consigers: true ,  erers: erss}
}else{
return {consigers: false ,  erers: 'Todo Correcto'}
}

}

Array.prototype.unique=function(a){
return function(){return this.filter(a)}}(function(a,b,c){return c.indexOf(a,b+1)<0
});

main();
```

Respuestas:

```
Validando parentesís...
¡Parentesis correctos!

Evaluando Operandos...
¡Operandos correctos!

Separando ecuacion...
[ '(','(','(','5','+','2','*','3','^','2',')','/','(','5','-
','3','/','(','8','-
','7',')',')',')','^','3','+','1',')','/','(','7','+','3','*','7',')',
'^','(','1','/','2',')' ]
¡Ecuacion separada!
```

```
Evaluando signos no permitidos...
¡No se detecto signo no permitido!

Pasando a Postfijo...
[ '5','2','3','2','^','*','+','5','3','8','7','-','/','-
','/','3','^','1','+','7','3','7','*','+','1','2','/','^','/' ]

Evaluando y resolviendo RPN...
[ 287.60734118045883 ]

FINALIZADO
```

Manual del usuario
El usuario puede resolver cualquier ecuación numerica que coloque, ya sea de multiples numeros, a prueba de errores o alguna prueba de crasheo.

Para que el programa funcione, se ejecuta el archivo rpn2.js, se le solicitara la ecuacion infija a resolver, el usuario coloca la ecuación y solo espera la respuesta.

En caso de que el usuario halla colocado algo mal, el algoritmo le mencionara su error.

Conclusiones

Conclusión grupal
Este proyecto a los 2 nos puso los cabellos verdes, pero a la vez nos hizo pensar, desarrollar y mejorar nuestros conocimientos y logica, para que podamos alcanzar el objetivo planteado, sin embargo de todo esto puedo decir que salio este proyecto, ¿Nos costo? Demasiado, al final logramos casi todos los objetivos y eso nos tiene satisfechos porque nos motivo a dar mas el siguiente proyecto a mirar al compañero y saber que no estas solo, tienes a alguien que te ayude cuando lo necesites.

Todos los conocimientos adquiridos en este proyecto siempre seran utiles para mas adelante, pilas, recursividad y manejo de errores, todo muy util para nuestra formacion.

Comentarios individuales

Hanna Lizarraga Ceballos
De este proyecto aprendí a usar estos algoritmos que sirven para identificar el significado de un texto y esto es relevante ya que todos los lenguajes de programación leen la sintaxis al programar y el procedimiento que usamos en este proyecto es similar. Me ayudo a comprender y a aprender a implementar de manera más compleja el uso de métodos. Al igual cabe recalcar que una vez más me di cuenta de que un simple y patético error en el código puede arruinar el funcionamiento de este y aún seguimos aprendiendo a ser más veloces a la hora de buscar errores, ya iremos mejorando en eso, en fin fue un proyecto algo complicado pero valió la pena después de todo el esfuerzo.

Manuel Jesús Canul Witzil

De este proyecto lo que mas pude aprender, fue de la forma en la que un algortimo me enseña como las cosas mas comunes que uso, estan hechas, el poder programar algo y sentirme muy orgulloso por eso es lo mejor que me pudo pasar durante este proyecto, al principio dude un poco de mis conocimientos, pero mientras programaba me di cuenta que si podia y pude, este es sin duda uno de los proyectos que se añaden a la lista de los proyectos que marcaron mi vida, porque mas alla de hacerme sufrir me enseñaron muchas cosas que podre usar durante mi carrea, como el uso de pilas, simular las pilas, comprender y sistematizar la seguridad de mis softwares y aprender el mejor manejo de errores.

Referencias

- Wikipedians, P. Algoritmos y Estructura de Datos.
- Jiménez Benito, R. (2014). *Mapas Veitch-Karnaugh en Android*(Bachelor's thesis).