PASO 3: ANTES

```
// MODULO DE CALCULADORA v2 - LEGACY
var buffer = "0";
var memoria = 0;
var ultimo_operador;
var historial = [];

function handleNumber(numStr) {
  if (buffer === "0") { buffer = numStr; } else { buffer += numStr; }
  updateScreen();
}

function handleSymbol(symbol) {
  switch (symbol) {
    case 'C':
      buffer = "0"; memoria = 0; ultimo_operador = null;
      break;
    case '=':
      if (ultimo_operador === null) { return; }
      // La funcion de calculo ahora tambien maneja el historial (MAL DISEÑO)
      flushOperationAndLog(parseInt(buffer));
      ultimo_operador = null;
      buffer = "" + memoria;
      memoria = 0;
      break;
    case '+': case '-': case '*': case '/':
      handleMath(symbol);
      break;
    // La logica cientifica (sin, cos, tan) no esta en el HTML, pero es codigo muerto
    case 'sin': case 'cos': case 'tan':
      if (buffer === "0") return;
      var cientifico_result;
      var val = parseFloat(buffer);
      if (symbol == 'sin') { cientifico_result = Math.sin(val); }
      else if (symbol == 'cos') { cientifico_result = Math.cos(val); }
      else if (symbol == 'tan') { cientifico_result = Math.tan(val); }
      buffer = "" + cientifico_result;
      // Logica de historial duplicada
      var logEntry = symbol + "(" + val + ") = " + cientifico_result;
      historial.push(logEntry);
      if (historial.length > 5) { historial.shift(); } // Magic Number!
      console.log(historial);
      break;
  }
  updateScreen();
}

function handleMath(symbol) {
  if (buffer === '0' && memoria === 0) { return; }
  var intBuffer = parseInt(buffer);
  if (memoria === 0) {
    memoria = intBuffer;
  } else {
    // La funcion de calculo ahora tambien maneja el historial (MAL DISEÑO)
    flushOperationAndLog(intBuffer);
  }
  ultimo_operador = symbol;
  buffer = "0";
}

// ESTA FUNCION HACE DEMASIADO (Long Method / Feature Envy)
function flushOperationAndLog(intBuffer) {
  var operacionPrevia = ultimo_operador;
  var memoriaPrevia = memoria;

  // Bloque 1: Calculo
  if (ultimo_operador === '+') { memoria += intBuffer; }
  else if (ultimo_operador === '-') { memoria -= intBuffer; }
  else if (ultimo_operador === '*') { memoria *= intBuffer; }
  else if (ultimo_operador === '/') { memoria /= intBuffer; }

  // Bloque 2: Logica de Historial (Duplicada y con Magic Number)
  var logEntry = memoriaPrevia + " " + operacionPrevia + " " + intBuffer + " = " + memoria;
  historial.push(logEntry);
  if (historial.length > 5) { historial.shift(); } // Magic Number y Duplicacion!
  console.log(historial); // Logica de UI mezclada con calculo
}

function updateScreen(){
  document.getElementById("display").innerText = buffer;
}

function init(){
  document.querySelector('.buttons').addEventListener('click', function(event) {
    buttonClick(event.target.innerText);
  });
}
function buttonClick(value) {
  if (isNaN(parseInt(value))) { handleSymbol(value); } else { handleNumber(value); }
}
init();
```

Métricas

Hay 8 funciones en este archivo.

Función con mayor toma de firma 1 argumentos, mientras que la mediana es 1.

La función más grande tiene 12 enunciados en él, mientras que la mediana es 3.5.

La función más compleja tiene un valor de complejidad ciclomática de 16 mientras que la mediana es 2.

PASO 3 : despues

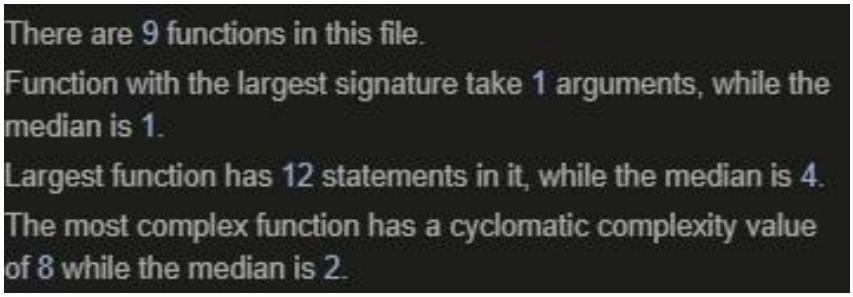
```
// MODULO DE CALCULADORA v2
var buffer = "0";
var memoria = 0;
var ultimo_operador;
var historial = [];

// REFACTOR (Linter Fix 1): 'const' cambiado a 'var' para
compatibilidad ES5.
var MAX_HISTORIAL_LENGTH = 5;
// REFACTOR (DRY): Función única para manejar la lógica del historial
function logToHistory(logEntry) {
  historial.push(logEntry);
  if (historial.length > MAX_HISTORIAL_LENGTH) {
    historial.shift();
  }
  console.log(historial);
}
function handleNumber(numStr) {
  if (buffer === "0") { buffer = numStr; } else { buffer += numStr; }
  updateScreen();
}
function handleSymbol(symbol) {
  // REFACTOR (Linter Fix 2): 'logEntry' se declara UNA VEZ aquí.
  var logEntry;
  switch (symbol) {
    case 'C':
      buffer = "0"; memoria = 0; ultimo_operador = null;
      break;
    case '=':
      if (ultimo_operador === null) { return; }
      // (Lógica de SRP sin cambios)
      var intBuffer = parseInt(buffer);
      var operacionPrevia = ultimo_operador;
      var memoriaPrevia = memoria;
      flushOperation(intBuffer);

      // REFACTOR (Linter Fix 2): Se quita 'var' de la asignación.
      logEntry = memoriaPrevia + " " + operacionPrevia + " " +
intBuffer + " = " + memoria;
      logToHistory(logEntry);
      ultimo_operador = null;
      buffer = "" + memoria;
      memoria = 0;
      break;
    case '+': case '-': case '*': case '/':
      handleMath(symbol);
      break;
```

```
    // REFACTOR (Dead Code):
    // El bloque 'case' para 'sin', 'cos', 'tan' ha sido eliminado.
    // Esto reduce la complejidad y limpia el código.
  }
  updateScreen();
}
function handleMath(symbol) {
  if (buffer === '0' && memoria === 0) { return; }
  var intBuffer = parseInt(buffer);
  if (memoria === 0) {
    memoria = intBuffer;
  } else {
      // (Lógica de SRP sin cambios)
      var operacionPrevia = ultimo_operador;
      var memoriaPrevia = memoria;
      flushOperation(intBuffer);

      // (Esta función está bien, 'var logEntry' solo se usa una vez
aquí)
      var logEntry = memoriaPrevia + " " + operacionPrevia + " " +
intBuffer + " = " + memoria;
      logToHistory(logEntry);
  }
  ultimo_operador = symbol;
  buffer = "0";
}
// (Resto de las funciones sin cambios)
function flushOperation(intBuffer) {
  if (ultimo_operador === '+') { memoria += intBuffer; }
  else if (ultimo_operador === '-') { memoria -= intBuffer; }
  else if (ultimo_operador === '*') { memoria *= intBuffer; }
  else if (ultimo_operador === '/') { memoria /= intBuffer; }
}
function updateScreen(){
  document.getElementById("display").innerText = buffer;
}
function init(){
  document.querySelector('.buttons').addEventListener('click',
function(event) {
    buttonClick(event.target.innerText);
  });
}
function buttonClick(value) {
  if (isNaN(parseInt(value))) { handleSymbol(value); } else
{ handleNumber(value); }
}
init();
```



Codigo de github

https://github.com/ndmb03/taller-2.git