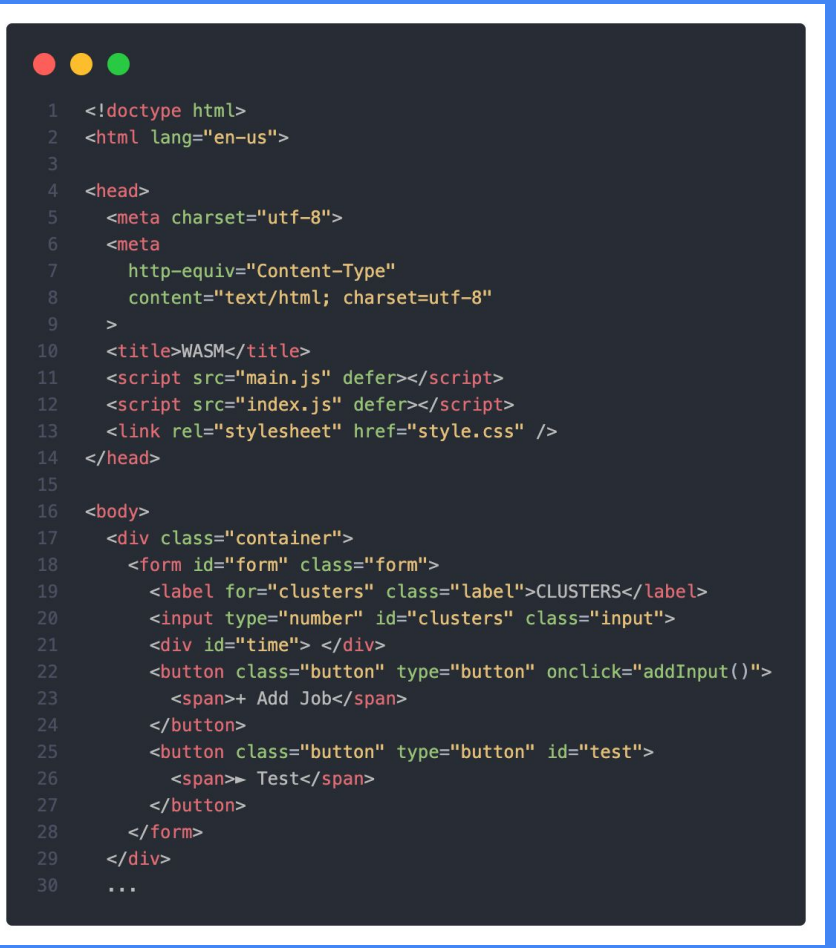
WebAssembly

Grupo 12

Daniela Castillo, Camila González, Benjamín Lillo

SOLUCIÓN CON EMSCRIPTEN

main.html



```
<!doctype html>
<html lang="en-us">

<head>
  <meta charset="utf-8">
  <meta
    http-equiv="Content-Type"
    content="text/html; charset=utf-8"
  >
  <title>WASM</title>
  <script src="main.js" defer></script>
  <script src="index.js" defer></script>
  <link rel="stylesheet" href="style.css" />
</head>

<body>
  <div class="container">
    <form id="form" class="form">
      <label for="clusters" class="label">CLUSTERS</label>
      <input type="number" id="clusters" class="input">
      <div id="time"> </div>
      <button class="button" type="button" onclick="addInput()">
        <span>+ Add Job</span>
      </button>
      <button class="button" type="button" id="test">
        <span>▸ Test</span>
      </button>
    </form>
  </div>
  ...
```

SOLUCIÓN CON EMSCRIPTEN

main.c

```c
void assignJobs(int nClusters, int nJobs, int* jobs, int** elements) {
  printf("1\n");
  // Input the time of each element and store it in the array
  for (int i = 0; i < nJobs; i++) {
    elements[i][0] = jobs[i];
    elements[i][1] = i; // Store the index of the element
    printf("%d, %d \n", elements[i][0], elements[i][1]);
  }

  // Sort the array of elements in non-decreasing order of time
  qsort(elements, nJobs, sizeof(int*), compareNumbers);

  // Assign each element to the group with the smallest sum
  int sum[nClusters];
  for (int i = 0; i < nClusters; i++){
    sum[i] = 0;
  }

  for (int i = 0; i < nJobs; i++) {
    int min_sum = sum[0];
    int min_index = 0;
    for (int j = 1; j < nClusters; j++) {
      if (sum[j] < min_sum) {
        min_sum = sum[j];
        min_index = j;
      }
    }
    sum[min_index] += elements[i][0];
    elements[i][0] = min_index; // Store the group of the element
  }

  for (int i = 0; i < nJobs; i++) {
    printf("%d, %d \n", elements[i][0], elements[i][1]);
  }
}
```

SOLUCIÓN CON EMSCRIPTEN

Llamado a funciones compiladas de c desde Javascript

```
emcc -o main.js main.c -s NO_EXIT_RUNTIME=1 -s "EXPORTED_RUNTIME_METHODS=['ccall']" -s EXPORTED_FUNCTIONS=_malloc,_free,_assignJobs
```

```
Module.onRuntimeInitialized = () => {
  document.getElementById('test').onclick = () => {
    const nClusters = parseInt(document.getElementById('clusters').value);

    const jobs = Array.from(document.getElementsByName('job')).map(element => parseInt(element.value));

    const jobsPtr = Module._malloc(jobs.byteLength);
    Module.HEAPU32.set(jobs, jobsPtr >> 2)

    // Allocate memory for the array of elements
    const elements = Module._malloc(jobs.length * 4);
    for (let i = 0; i < jobs.length; i++) {
      elements[i] = Module._malloc(2 * 4);
    }

    const result = Module.ccall('assignJobs', null,
                   ['number', 'number', 'number', 'number'],
                   [nClusters, jobs.length, jobsPtr, elements]);

    const elementsArray = new Int32Array(Module.HEAPU32.buffer, elements, jobs.length);

    console.log(`elementsArray: ${elementsArray}`);

    Module._free(jobsPtr);

  };
};
```

SOLUCIÓN CON EMSCRIPTEN

Llamar JavaScript desde c

```
double fin = clock();

EM_ASM(
      {
        document.getElementById('duracion-c')
          .innerHTML = $0;
        document.getElementById('loader-duracion-c')
          .style.display = 'none';
      },
      (fin - inicio) / CLOCKS_PER_SEC
);
```

Algoritmos implementado

Algoritmo en C

```c
void assignJobs(int nClusters, int nJobs, int* jobs) {
  double inicio = clock();

  qsort(jobs, nJobs, sizeof(int*), compareNumbers);


  int elements[nJobs][2];
  for (int i = 0; i < nJobs; i++) {
    elements[i][0] = jobs[i];
    printf("%d\n", elements[i][0]);
  }

  int sum[nClusters];
  for (int i = 0; i < nClusters; i++){
    sum[i] = 0;
  }

  for (int i = 0; i < nJobs; i++) {
    int min_sum = sum[0];
    int min_index = 0;
    for (int j = 1; j < nClusters; j++) {
      if (sum[j] < min_sum) {
        min_sum = sum[j];
        min_index = j;
      }
    }
    sum[min_index] += elements[i][0];
    elements[i][1] = min_index;
  }

  printf("SOLUCIÓN: \n");
  for (int i = 0; i < nJobs; i++) {
    printf("Tiempo: %d, cluster: %d \n", elements[i][0], elements[i][1]);
  }

  double fin = clock();

  EM_ASM(
      {document.getElementById('duracion-c').innerHTML = $0;
      document.getElementById('loader-duracion-c').style.display = 'none';
      }, (fin - inicio) / CLOCKS_PER_SEC);
}
```

Algoritmo en JS

```javascript
const jsSolution = (N, M, times) => {
  const MAX_N = 1000
  const MAX_M = 1000

  // obtenido de https://stackoverflow.com/questions/5185864/javascript-quicksort
  function quicksort(array) {
    if (array.length <= 1) {
      return array;
    }

    var pivot = array[0];

    var left = [];
    var right = [];

    for (var i = 1; i < array.length; i++) {
      array[i][1] < pivot[1] ? left.push(array[i]) : right.push(array[i]);
    }
    return [...quicksort(left), pivot, ...quicksort(right)]
  };

  let sum = Array.from({length: MAX_M}, (v, i) => 0);
  let elements = Array.from({length: N}, (v, i) => [0, 0]);

  for (let i = 0; i < N; i++) {
    elements[i][0] = times[i];
    elements[i][1] = i;
  }

  elements = quicksort(elements)

  for (let i = 0; i < N; i++) {
    let min_sum = sum[0];
    let min_index = 0;
    for (j = 1; j < M; j++) {
        if (sum[j] < min_sum) {
            min_sum = sum[j];
            min_index = j;
        }
    }
    sum[min_index] += elements[i][0];
    elements[i][0] = min_index; // Store the group of the element
  }

  // Print the group of each element
  const res = elements.map((element => element[0]));
```

WebAssembly

Grupo 12

Daniela Castillo, Camila González, Benjamín Lillo