Цель работы:

Научиться создавать, управлять и использовать строки и двоичные данные в WebAssembly, а также интегрировать их с JavaScript.

Инструкции:

1. **Создайте файл WAT для каждой задачи.**
2. **Скомпилируйте WAT-файлы в WASM с помощью утилиты `wat2wasm`.**
3. **Разработайте HTML и JavaScript для загрузки и использования WASM-модулей.**
4. **Запустите код в браузере и проверьте результаты.**

Задание 1: Строки с завершающим нулем

Вариант 1:

1. Создайте WAT-файл, который определяет строку `"HelloWorld"` с завершающим нулем в памяти.
2. Реализуйте функцию, которая возвращает адрес строки в памяти.
3. В JavaScript загрузите WASM, получите строку по её адресу и выведите её в консоль.

Вариант 2:

1. Определите строку `"WebAssembly"` с завершающим нулем в WAT-файле.
2. Реализуйте функцию для получения длины строки (до завершающего нуля).
3. Используйте JavaScript для получения строки по адресу и её длины, затем выведите строку и её длину.

Вариант 3:

1. Создайте строку `"StringTest"` с завершающим нулем в WAT-файле.
2. Реализуйте функцию для замены символа в строке (например, заменить первый символ на `'*'`).
3. Используйте JavaScript для изменения строки и вывода её результата.

Вариант 4:

1. Определите строку `"InitialString"` с завершающим нулем в WAT-файле.
2. Реализуйте функцию для нахождения индекса символа (например, индекс символа `'S'`).
3. В JavaScript выведите найденный индекс и саму строку.

Задание 2: Копирование строк

Вариант 1:

1. Определите строку `"SourceString"` в WAT-файле.
2. Реализуйте функцию, которая копирует эту строку в новый адрес памяти.

3. В JavaScript скопируйте строку и выведите обе строки (исходную и скопированную).

Вариант 2:

1. Создайте строку `"OriginalData"` и определите её размер.
2. Реализуйте функцию, которая копирует строку с учётом размера и завершающего нуля.
3. В JavaScript выведите оригинальную и скопированную строки, а также их адреса в памяти.

Вариант 3:

1. Определите строку `"CopyThis"` в WAT-файле и выделите новый участок памяти для копии.
2. Реализуйте функцию для копирования строки в новый адрес памяти.
3. В JavaScript выведите оригинальную строку и строку, находящуюся в новом месте в памяти.

Вариант 4:

1. Создайте строку `"DataToCopy"` и определите её длину.
2. Реализуйте функцию, которая копирует данные, игнорируя завершающий нуль, в новое место памяти.
3. Используйте JavaScript для проверки корректности копирования строки.

Задание 3: Создание двоичных строк

Вариант 1:

1. Определите двоичную строку (массив байтов) в WAT-файле, например, `"\01\02\03\04\05"`.
2. Реализуйте функцию для получения и вывода этой двоичной строки.
3. В JavaScript выведите содержимое двоичной строки в консоль.

Вариант 2:

1. Создайте двоичные данные в WAT-файле: `"\10\20\30\40\50"`.
2. Реализуйте функцию, которая возвращает длину этих двоичных данных.
3. В JavaScript прочитайте и выведите длину данных и их содержимое.

Вариант 3:

1. Определите двоичную строку `"\AA\BB\CC\DD"` в WAT-файле.
2. Реализуйте функцию для обработки данных (например, сложение всех байтов).
3. В JavaScript вызовите эту функцию и выведите результат обработки.

Вариант 4:

1. Создайте двоичные данные `"\01\FF\AB\CD"` в WAT-файле.
2. Реализуйте функцию для замены одного из байтов (например, заменить `\FF` на `\00`).

3. В JavaScript выведите изменённые двоичные данные и их длину.

Примеры выполнения:

Пример создания WAT и JavaScript:

1. **WAT-файл (example.wat):**

```
wat
Копировать код
(module
  (memory (export "memory") 1)
  (data (i32.const 0) "HelloWorld\00")

  (func (export "get_string_pointer") (result i32)
    (i32.const 0)
  )
)
```

2. **Компиляция WAT в WASM:**

```
bash
Копировать код
wat2wasm example.wat -o example.wasm
```

3. **JavaScript (index.html):**

```
html
Копировать код
<!DOCTYPE html>
<html>
<body>
  <script>
    async function loadWasm() {
      const response = await fetch('example.wasm');
      const wasmArrayBuffer = await response.arrayBuffer();
      const wasmModule = await
WebAssembly.instantiate(wasmArrayBuffer);

      const { memory, get_string_pointer } =
wasmModule.instance.exports;
      const memoryBuffer = new Uint8Array(memory.buffer);
      const stringPointer = get_string_pointer();
      const stringEnd = memoryBuffer.indexOf(0, stringPointer);
      const stringData = new
TextDecoder().decode(memoryBuffer.slice(stringPointer, stringEnd));

      console.log('String from WASM memory:', stringData);
    }

    loadWasm();
  </script>
</body>
</html>
```