Лабораторная 3

Задание 1

1. Состояние приложение переключается с помощью стандартного системного ввода, который назначен на консоль. Консоль в приглашении (prompt) указывает текущее состояние приложения.

2. Пользователь может ввести новое состояние (norm, stop, test, idle). При корректном вводе состояния осуществляется переключение состояния приложения.

3. При ошибочном вводе режима, ошибочная введенная последовательность символов просто отображается, но переключение режима не осуществляется.

4. Допускается ввод состояния exit, которое приводит к завершению приложения. **prompt** - Что вывести на экран (stdout) перед чтением ввода (stdin).

Задание 2

1. Для расчета факториала используйте рекурсивный алгоритм.

В языках программирования рекурсивной программой называется программа, которая обращается сама к себе (подобно тому, как в математике рекурсивная функция определяется через понятия самой этой функции). Рекурсивная программа не может вызывать себя до бесконечности, следовательно, вторая важная особенность рекурсивной программы – наличие условия завершения, позволяющее программе прекратить вызывать себя.

Задание 4

**process.nextTick** - срабатывает сразу на той же фазе

Задание 5

**setImmediate** - предназначен для выполнения сценария после завершения текущей фазы опроса.

# 

# 1. Перечислите основные свойства глобальные объекты Node.js и поясните их предназначение.

**global** - хранит var-данные на уровне модуля

**process** - информация о среде выполнения, о текущем процессе

**buffer** - класс Buffer - предназначен для работы с двоичными данными

**console** - используется для печати в stdout и stderr

Существуют только в составе модулей:

**exports** - ссылка на module.exports

**module** - ссылка на текущий модель

**require** - используется для импорта модулей

# 2. Поясните понятие «асинхронная функция».

**Асинхронная функция** - это функция, после вызова которой JS приложение продолжает работать, потому что функция сразу выполняет возврат

# 3. Поясните понятие стандартные «системные потоки».

Эти потоки поставляются с node.js и могут быть использованы без подключения дополнительных библиотек

**process.stdin** - поток на чтение содержит стандартный системный поток ввода.

**process.stdout** - поток на запись, содержащий стандартный системный вывод.

**process.stderr** - поток на запись, содержащий стандартный системный вывод ошибок.

# 4. Поясните назначение функций **process.nextTick**, **setImmediate,** поясните в чем разница.

**setImmediate()** - преданный callback будет выполнен на следующей итерации цикла событий, callbak’и перемещаются в цикл обработки событий

**nextTick()** - переданный callback бедет выполнен после предыдущей итерации цикла событий и до следующей колбэки не помещаются в цикл обработки событий

**setImmediate()** является обратным вызовом, который выполняется на следующей итерации цикла событий.

**process.nextTick()** будет выполняться на текущей итерации цикла событий после завершения текущей операции.

--------------Лабораторная 4

# 1. Дайте пояснению понятию «событие программного объекта».

Процесс перехода объекта из одного состояния в другое.

# 2. Объясните механизм генерации и обработки событий в C#.

- Описание делегата, задающего сигнатуру обработчиков событий

- Описание события

- Описание методов, инициирующих событие

# --3. Поясните как самостоятельно реализовать механизм генерации и обработки событий на JS или С++.

# 4. Какой встроенный механизм используется в **Node.js** для генерации и обработки событий. Поясните принцип его работы.

**EventEmitter:** JS-класс, предоставляющий функциональность для асинхронной

обработки событий в Node.js.

Лабораторная 5

## 1. setTimeout

позволяет вызвать функцию один раз через определенный интервал времени.

## 2. setInterval

позволяет вызывать функцию регулярно, повторяя вызов через определённый интервал времени.

## 3. ref

Уберет изменения внесенные unref

## 4. unref

изменяет поведение Timeout(возвращается функциями выше). Timeout не выполнит запланированный код, если это последний код, который надо выполнить