Написать диагностику, которая будет искать ошибку в исходном коде на языке JavaScript, используя Esprima.

## Вводная часть

Компания PVS-Studio занимается разработкой одноименного статического анализатора. Одним из основных направлений развития анализатора является реализация новых интересных и нетривиальных диагностик. Новые диагностики очень важны, так как они позволяют находить новые ошибки. Поэтому наше задание связано именно с написанием диагностики =)

Чтобы вам было с чего начать, мы подготовили для вас [мини-проект](https://github.com/PVSStudioHacks/TulaHack2021/blob/main/sources/examples/Esprima.7z), который будет содержать минимум того, чтобы вы сразу приступили к написанию диагностического правила. Так же в нем будет приведен пример, для поиска следующей ошибки. В файле Project.js имеется вот такой код:

1. function Main(param) {

2. let i = param;

3. i = i++;

4. }

....

В результате работы диагностического правила будет выдано предупреждение на строку 3 файла Project.js, указывающее на то, что выполнение постфиксного инкремента для переменной ‘i’ бессмысленно, потому что эта переменная сразу же перезаписывается.

## О задании

Нужно реализовать диагностику для поиска ошибки в исходном JavaScript коде, на основe Esprima .NET. В результате ее работы должно находиться сообщение об ошибке, место ошибки (файл, строка) и номер диагностики. Выводить можно сообщение об ошибке можно, например, в текстовый файл.

Esprima – это парсер JavaScript кода, который служит для получения [AST](https://www.viva64.com/ru/t/0004/) (Abstract Syntax Tree). Подробнее можно почитать [тут](https://esprima.org/).

Примеры диагностических правил для реализации (**на выбор**):

**1.** Имя переменной перепутали со строковым литералом, который содержит в себе имя переменной.

function Error1(userName) {

let name = GetStrValue();

if (userName == "name") { // <=

// Do something

}

}

**2.** Cчётчик цикла запустили не в ту сторону.

function Error2(content) {

for (i = 0; i < content.length; i--) {// <=

// Do something

}

}

**3.** Сравнивают не то, что подразумевалось. Сopy-paste ошибка.

function Error3(a, b) {

if (a.X == b.X && a.Y == b.Y && a.Z == b.Y) { // <=

// Do something

}

}

**Советы:**

* К реализации диагностики подходите с точки зрения TDD (сначала тесты, потом реализация)
* Диагностические правила для полученной метамодели являются визиторами, поэтому знание о таком паттерне вам может [помочь](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%B5%D1%82%D0%B8%D1%82%D0%B5%D0%BB%D1%8C_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)#Java).
* Вам так же вам поможет помочь [онлайн демонстрация перевода](https://esprima.org/demo/parse.html) исходного кода в [AST](https://www.viva64.com/ru/t/0004/).

**Приветствуется:**

* Развитие удобной инфраструктуры (многопоточный анализ, походы ООП  и прочее)
* Реализация как можно большего количества диагностических правил
* Реализация своих диагностических правил