# Архитектура алгоритма

Далее будет описан алгоритм анализа бесконечного входного видеопотока (последовательность изображений) с целью поиска в нем любых частей из массива заранее подготовленных роликов. Для достижения поставленной цели, необходимо создать базу роликов, в которой будет производиться поиск.

Каждый ролик разделяется на сегменты похожих друг на друга кадров. Например, когда в фильме начинается новая сцена, то первый кадр новой сцены сильно отличается от предыдущего кадра (последнего кадра предыдущей сцены). Далее в фильме могут идти изображения, похожие на первый кадр в сцене. Соответственно, при сравнении входного изображения с кадрами ролика можно ограничить сравнение только с первыми кадрами таких сегментов. Эту сильно ускорит процесс сравнения текущего изображения со всеми роликами в базе данных. При этом нужно сравнивать сигнатуры достаточно грубо (с большим порогом сравнения), чтобы отсеять подавляющее большинство ненужных роликов. А затем можно провести более точное сравнение с очень маленьким числом похожих роликов.

Далее будут описаны структуры хранения частей ролика, а также алгоритм их наполнения. Для работы с сигнатурами, были введены 3 основных порога сравнения.

Основные понятия и константы:

* VideoPart – структура хранения части ролика.

* – порог резкой смены сцены.

* – порог сжатия. Данный порог необходимо вычислить эмпирически, путем сравнения исходного видео и после различных популярных алгоритмов сжатия.

* – порог сцены. Данный порог настраивается в зависимости от категории видео. Так же данный порог подается в алгоритм поиска. Ориентировочные характеристики: , n = 2,3,4,5…
* mainSignature – сигнатура, являющаяся представителем ячейки VideoPart.

## Структуры и классы для хранения и работы с информацией

## Класс для хранения сигнатур ролика (**SignatureHandler**), а также его подготовленных частей (**VideoPart**)

При подготовке ролика для хранения в базе данных, первым делом необходимо вычислить сигнатуры всех его кадров. За хранение набора сигнатур, вычисленных из кадров подготавливаемого ролика, отвечает класс **SignatureHandler**. Для хранения и работы с сигнатурами был выбран массив как структура данных. Работа с сигнатурами будет производиться последовательно, а сам массив статичным. В этом случае, массив предоставляет наибыстрейший последовательный перебор элементов, а статичность данных позволяет избежать дорогостоящей операции увеличения размеров массива. Доступ к сигнатуре будет производиться по индексу, что массив как структура выполняет за константное время - O(1).

## Структура хранения части ролика: “VideoPart”

Разбиение ролика на части происходит по двум основным принципам. Первый основывается на том, что соседние кадры с большой разницей сигнатур – означают различные сцены (порог C\_1). Высокий порог при смене сцены, позволяет с большой точностью утверждать, что часть, найденная до смены сцены является искомой. Второй принцип основывается на наблюдениях особенности видеороликов. Довольно часто бывает такая ситуация, что кадры сменяются плавно, например от цветного к черному или наоборот. При такой ситуации сравнение соседних сигнатур, не достигает порога С\_1, что приводит к большому размеру **VideoPart**, а как следствие к ошибкам в работе алгоритма поиска совпадающих частей. Что бы избежать такой ситуации вводится второй критерий: сравнение первой и последней сигнатуры в **VideoPart**. Если это сравнение больше порога C\_3, **VideoPart** заканчивается и начинается следующий, этот прием ускорит отсеивание ошибочно найденных **VideoPart**, а также увеличит точность алгоритма поиска похожих частей. После определения границ **VideoPart**, записываются индексы граничных сигнатур.

Далее будут описаны поля структуры хранения части ролика:

* **mainSignatureIndex** - номер ключевого кадра. Ключевой кадр, это кадр подготавливаемого ролика, который был выбран как основной для текущей VideoPart.
* **lastSignatureIndex** **-** bндекс последнего кадра, входящего в промежуток текущего VideoPart. (lastFrameIndex)
* **selfIndex** – собственный индекс в классе **signatureHandler.**

## Структура для хранения размера поискового окна: “**SearchWindow**”

После получения кадра из видеопотока, его сигнатура подается на вход алгоритму поиска. Алгоритм поиска выполняет сравнение данной сигнатуры с ключевыми сигнатурами VideoPart, но с достаточно высоким порогом (больше порога С\_2). Предполагается что данный порог будет равен С\_3. Это сделано, чтобы избежать ситуации, когда полученный кадр из видео потока не является началом **VideoPart**, ведь он может находится внутри **VideoPart. VideoPart** устроен так, что в нем находятся похожие кадры, в связи с этим не всегда можно утверждать что найденный кадр и естб искомый, возможны ситуации когда таких кадров несколько, в связи с этим возникает необходимость создания “окна” поиска – **SearchWindow**. Данная структура, хранит данные о размере окна. Так же к отдельной категории относятся ситуации пропуска кадра в входящем видеопотоке, для этого, стандартным размером окна является +- 1 кадр, от кадра с которым ведется работа. В других случаях размер может быть произвольным.

Поля структуры **SearchWindow**:

* **leftIndex** – абсолютный индекс левого края окна.
* **rightIndex** – абсолютный индекс правого края окна.

## Структура для хранения данных о поиске: “**SearchInfo**”

Алгоритм поиска похожих чатей по сигнатуре входящего видео с порогом C\_3, может возвращать несколько таких частей, так как каждую из них необходимо обработать, так же продолжать работать на нескольких итерациях сравнения, возникает потребность работы с несколькими **VideoPart** одновременно. Структура **SearchInfo** созданная решить эту проблему, объединяя в себе всю необходимую информацию для работы.

Поля структуры **SearchInfo:**

* + **searchWindow** – размеры окна с которым ведется работа.
  + **videoPart** – непосредственно **VideoPart** с которым ведется работа.
  + **sigIndex** – текущий индекс в массиве сигнатур из **SignatureHandler**, на который указывает VideoPart.
  + **isWorkWith** – булевская переменна, отвечающая за то, что мы уже работали с этим блоком.

Для хранения нескольких **SearchInfo**, был выбран список как структура данных. Данная структура, обладает достаточными по скорости характеристиками в рамкой поставленной задачи.

## Структура, представляющая из себя: ключ/значение для создания структуры поиска: “SearchCell”.

Для создания базы данных по подготовленным роликам, информации из **VideoPart** недостаточно, а также необходимо привести ее в общепринятый вид (ключ/значение). Так же стоит учитывать, что база данных, должна иметь возможность сохранять информацию на диск, а также быть интерпретируемой в различных алгоритмах. Для этого были выбраны индексы, как основной инструмент взаимодействия с базой данных.

Ключ:

Ключом для алгоритма построения базы данных будет являться **mainSignatureIndex** из **VideoPart**.

* **signatureIndex** – индекс сигнатуры, которая учувствует в построении структуры поиска.

Значение:

* **signatureHandlerIndex** – индекс класса **SignatureHandler**, в массиве всех роликов (**DataInformation**). В этом **SignatureHandler** находится сигнатура по индексу: signatureIndex.
* **videoPartIndex** – индекс класса **VideoPart** в массиве videoParts, который относится к **SignatureHandler** по индексу signatureHandlerIndex. Так же в этом **VideoPart** сигнатура по индексу **signatureIndex**, является **mainSignature**.

## Класс, отвечающий за хранение информации обо всех роликах: “DataInformation”

Данный класс, призван объединить все подготовленные ролики, в единую базу. Так же **DataInformation** выполняют важную функцию подготовки имеющейся информации для создания базы поиска, а также структурирования информации. Для этого создается массив структур **SearchCell**.

Данный класс отвечает за сохранения информации о роликах в памяти долгого хранения, а также чтения этой информации и восстановления информации о базе из памяти.

## Алгоритмы

## Алгоритм сравнения кадра из потока видео с массивом VideoPart: **“PartSearcher”**.

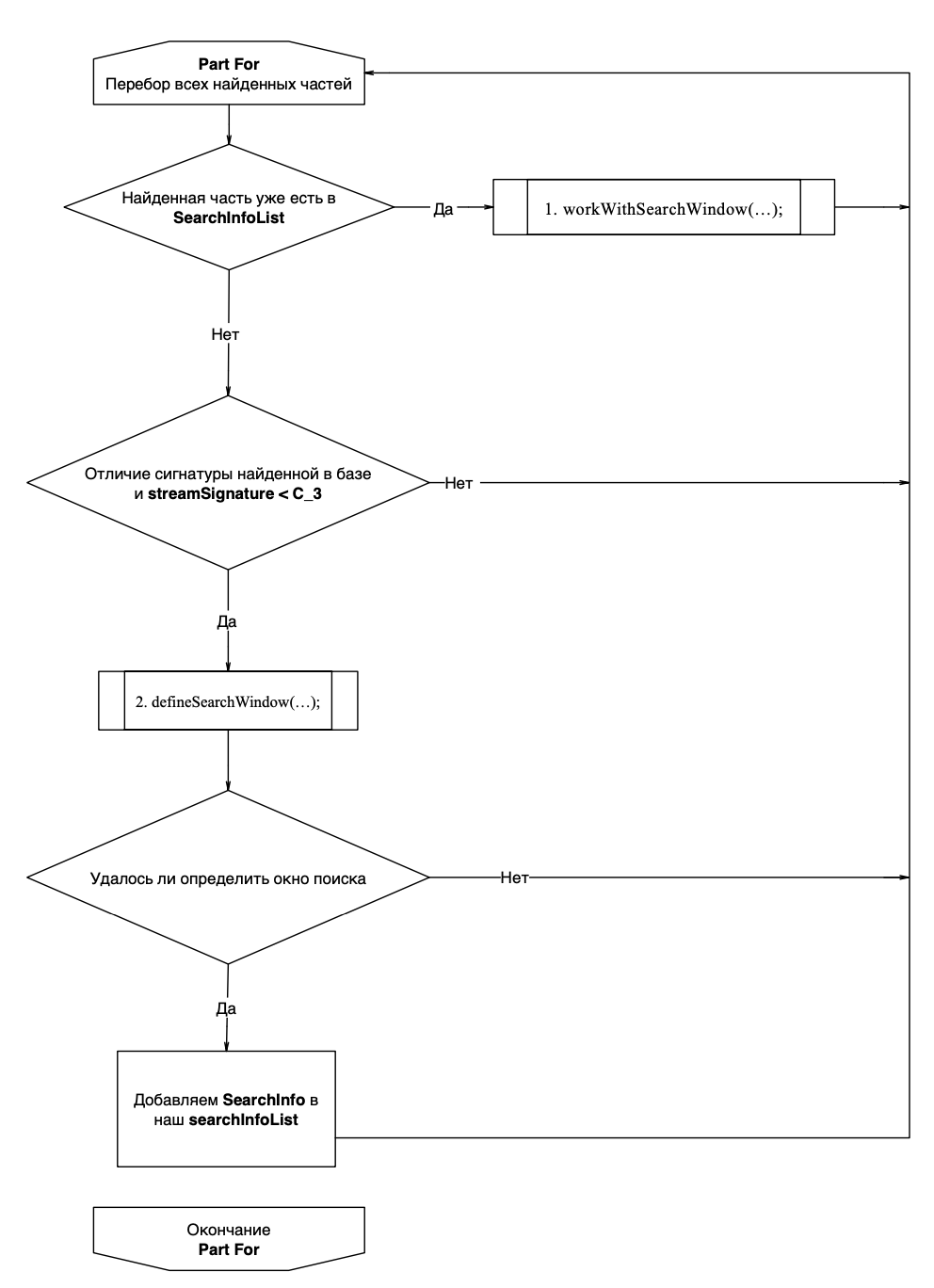
**PartSearcher** – класс отвечающий за поиск совпадающих частей из входящего видеопотока и структуры хранения данных.

## StartSearching()

Данная функция является StateMachine. Берет кадр из потока и начинает свою работу, пока в потоке есть кадры.

По полученному кадру выполняется поиск похожих частей с порогом C\_3. Если похожие части найденные, выполняется алгоритм, представленный на рисунке ниже.

* **streamSignature** - сигнатура текущего кадра из видеопотока,
* **searchInfoList** – лист из экземпляров класса **SearchInfo**.

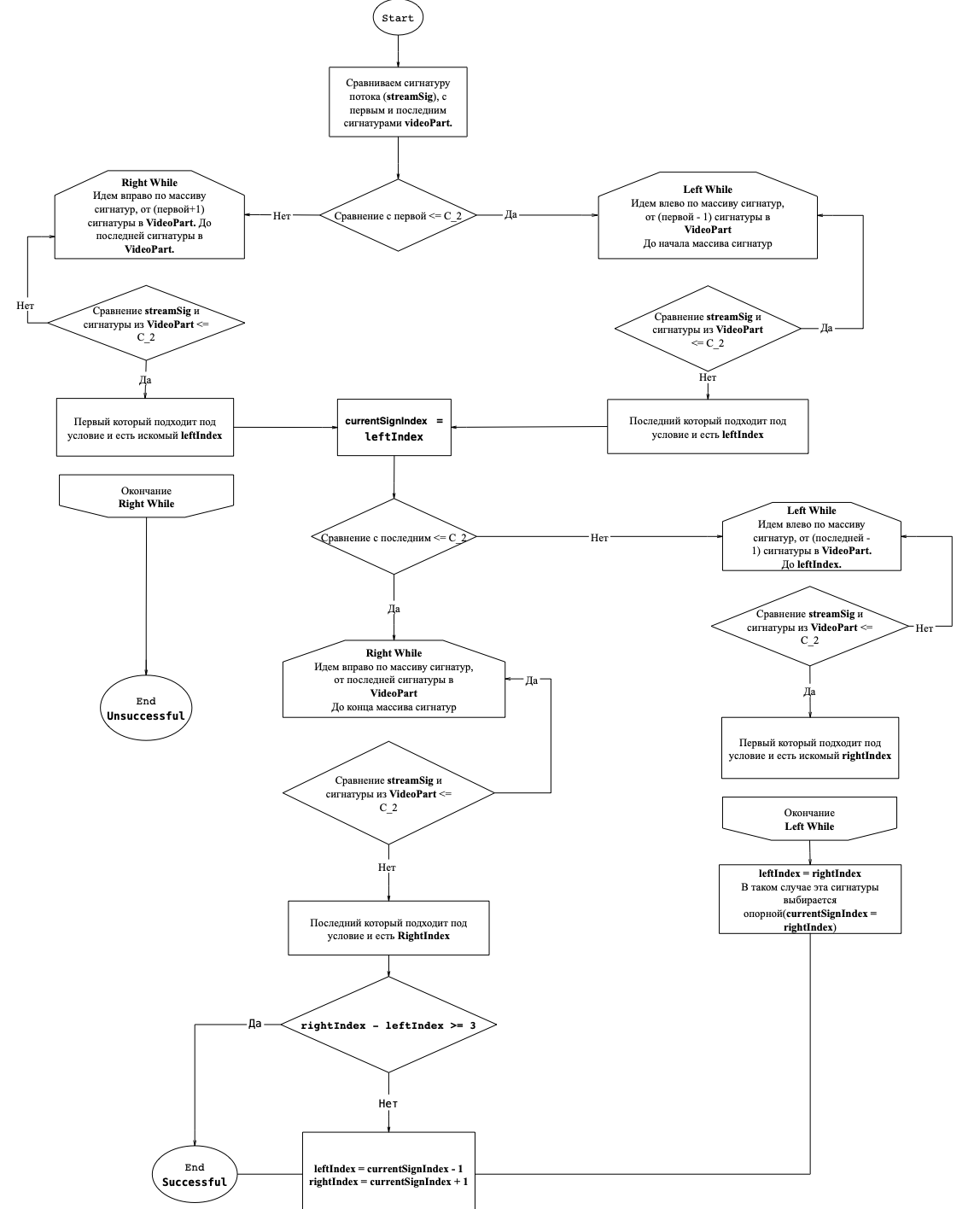


После окончания работы с найденными частями, выполняется повторный   
проход по **SearchInfoList** и обработка тех **SearchInfo**, с которыми работа   
не производилась, функцией **workWithSearchWindow**.

## defineSearchWindow(SearchInfo &currentSearchInfo, CSignature \*streamSig)

Данная функция позволяет выбрать кадр, с которого начать работать с текущим **VideoPart**, а также определить окно поиска в рамках текущего **VideoPart.**

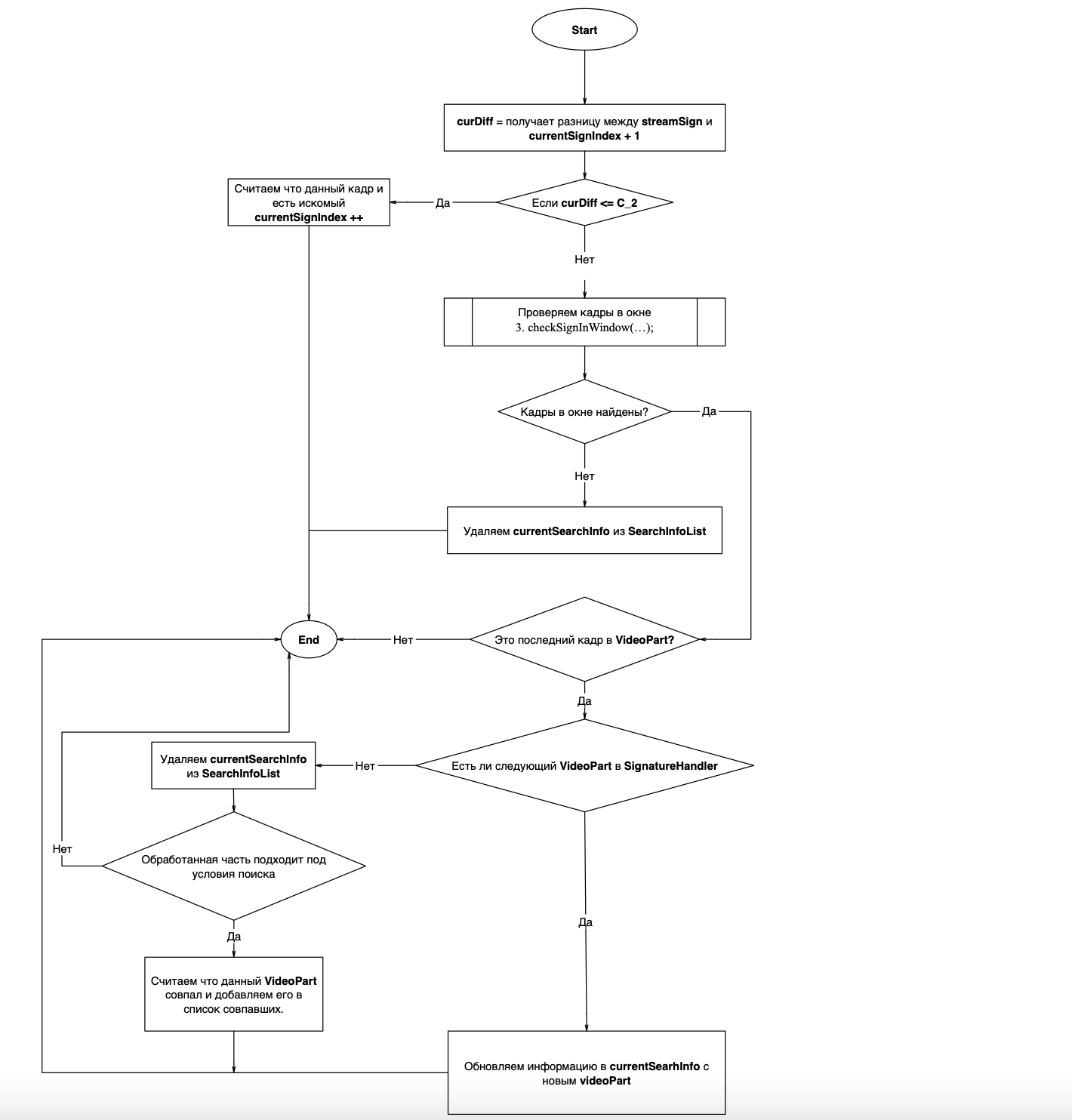
* **leftIndex** – абсолютный индекс сигнатуры в массиве сигнатур ролика. Обозначает левый край окна поиска.
* **rightIndex** - абсолютный индекс сигнатуры в массиве сигнатур ролика. Обозначает правый край окна поиска.
* **currentSignIndex** – индекс опорного кадра. Предполагается что именно этот кадр и есть искомый (или с него нужно начать поиск). Указывает на сигнатуру в массиве сигнатур.



## workWithSearchInfo(SearchInfo &currentSearchInfo, CSignature \*streamSign)

Данная функция выполняет работу с **SearchInfo**. Берет сигнатуру из **VideoPart** и считает разницу сигнатур. В зависимости, от которой выполняет те или иные действия.

* **currentSignIndex** - индекс текущего кадра в **VideoPart**.
* **currentSearchInfo** – класс **SearchInfo** с которым ведется работа.



## checkSignInWindow(SearchInfo &currentSearchInfo, CSignature \*streamSig)

Данная функция выполняет корректировку окна (если это необходимо), а также меняет индекс опорного кадра.

* **currentSearchInfo** - класс **SearchInfo** с которым ведется работа,
* **streamSig** – сигнатура из видеопотока,
* **leftIndex** - левый край окна в массиве сигнатур ролика,
* **rightIndex** - правый край окна в массиве сигнатур ролика,
* **currentSignIndex** - индекс ожидаемого кадра в массиве сигнатур ролика.

