Департамент образования города Москвы

Государственное бюджетное общеобразовательное учреждение

города Москвы

«Гимназия № 1505 «Московская городская педагогическая гимназия-лаборатория»»

**ДИПЛОМНОЕ ИССЛЕДОВАНИЕ**

на тему:

**Система биометрической идентификации**

Выполнил (а):

Акинина Софья Андреевна 10 «Б» класс

Руководитель

Наумов Алексей Леонидович

подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент:

Голодняк Михаил Михайлович

подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

 2017/2018 уч.г.

Оглавление

Введение 3

Первая глава 4

1.1 Идентификация пользователя 4

1.2 Механизм идентификации пользователя 5

1.3 Идентификация по биометрическим признаками (физическим) 5

1.3.1 Отпечатки пальцев 5

1.3.2 Радужная оболочка глаза 6

1.3.3 Распознавание лица 7

1.3.4 Распознание по венам рук 8

1.4 Идентификация по клавиатурному почерку 9

Вторая глава 11

1. Программа для регистрации нажатий клавиш 11

2. Программа для выделения смежных букв 11

3. Программа для сортировки буквосочетаний 12

4. Программа для нахождения медиан 13

5. Программа для выделения почерка 14

6. Программа для проверки подлинности 16

Заключение 17

Список литературы 18

Приложения 19

# Введение

Каждый человек имеет свои особенности, которые являются его биометрическими характеристиками. Биометрические характеристики индивидуальны у каждого человека, поэтому растет интерес к идентификации с использованием этих характеристик. В качестве критерия биометрической идентификации я предлагаю использовать особенности, связанные с клавиатурным вводом. Данный способ идентификации является программным и не требует специальных устройств, таких как сканер отпечатков пальцев. Поэтому он может применяться скрытно, незаметно для пользователя, что дает возможность также его использовать для обнаружения несанкционированного доступа.

Традиционные методы идентификации по паролю или коду доступа недостаточно достоверны и надежны. Пароль можно украсть, подсмотреть или просто подобрать. А биометрические характеристики сопутствуют человеку всю его жизнь.

В моем исследовании я собираюсь написать программу, собирающую статистические данные, а так же программу для их обработки. После чего я хочу проанализировать собранные данные, выявить почерки нескольких пользователей и проверить, насколько точно возможно опознать человека по его почерку.

# Первая глава

## 1.1 Идентификация пользователя

Идентификация – это присвоение субъектам или объектам личного идентификатора и сравнение его с заданным. Идентификация пользователя позволяет выяснить, является ли пользователь тем, за кого он себя выдает.

Существует много вариантов идентификаторов пользователей. В целом их можно разделить на две группы:

* Набор символов ( пароль, секретный ключ, персональный идентификатор и т.д), который пользователь запоминает и вводит при необходимости идентификации
* Биометрические характеристики: физические параметры человека или характеристики, связанные с поведением

Наиболее привычным в наши дни является использование пароля. Парольные методы можно разделить по степени изменяемости паролей на:

* Методы, использующие постоянные пароли
* Методы, используемые одноразовые пароли

Использование одноразовых паролей является более надежным.

Но в наше время, все больше и больше популярности и актуальности набирают методы идентификации, связанные с биометрическими параметрами:

* Проверка физических параметров (отпечатки пальцев, рисунок радужной оболочки глаза)
* Проверка поведенческих характеристик (особенности клавиатурного почерка и т.д)

## 1.2 Механизм идентификации пользователя

Пользователь предоставляет системе свой личный идентификатор (например, вводит пароль или предоставляет палец для сканирования отпечатка). Далее система сравнивает полученный идентификатор со всеми хранящимися в ее базе идентификаторами, хранящимися в ее базе. Если результат сравнения успешный, то пользователь получает доступ к системе в рамках установленных полномочий. В случае отрицательного результата система сообщает об ошибке и предлагает повторно ввести идентификатор. В тех случаях, когда пользователь превышает лимит возможных повторов ввода информации (ограничение на количество повторов является обязательным условием для защищенных систем) система временно блокируется и выдается сообщение о несанкционированных действиях (причем, может быть, и незаметно для пользователя).

В целом, алгоритм работы систем идентификации заключается в получении от субъекта или объекта информацию, удостоверяющую личность и, проверив ее подлинность, разрешении доступа или нет.

## 1.3 Идентификация по биометрическим признаками (физическим)

### 1.3.1 Отпечатки пальцев

Дактилоскопия (распознание отпечатков пальцев) является одним из самых разработанных и популярных биометрических методов распознавания личности.

Каждый человек имеет свой собственный папиллярный рисунок, что позволяет возможна почти достоверная идентификация. Система использует характерные точки на отпечатках пальцев: окончание линий узора, разветвления и одинокие точки. После сканирования отпечаток преобразовывается в уникальный код. И именно коды отпечатков пальцев хранятся в базе данных.

Преимущества метода:

* Достоверность, по сравнению с идентификацией по голосу или лицу
* Низкая стоимость устройств, сканирующих отпечатки пальцев
* Простая процедура сканирования

Недостатки метода:

* Папиллярный рисунок очень легко повреждается мелкими царапинами или порезами
* Присутствует недостаточная защищенность, связанная с широким распространением метода

### 1.3.2 Радужная оболочка глаза

Радужная оболочка глаза является уникальной характеристикой человека. Она стабилизируется в возрасте около двух лет и не практически не меняется на протяжении всей жизни. Следовательно этот метод является одним из наиболее точных среди биометрических методов.

Система идентификации личности по радужной оболочке логически делится на две части: устройство захвата изображения, его первичной обработки и передачи вычислителю и вычислитель, производящий сравнение изображения с изображениями в базе данных, передающий команду о допуске исполнительному устройству.

Преимущества метода:

* Статистическая надежность алгоритма.
* Захват изображения радужной оболочки можно производить на расстоянии от нескольких сантиметров до нескольких метров.
* Радужная оболочка не изменяется на протяжении жизни, за исключением травм или патологий

Недостатки метода:

* Цена системы
* Низкая доступность готовых решений

### 1.3.3 Распознавание лица

2-D распознавание является статистически неэффективным методом. Появился он достаточно давно и применялся, в основном в криминалистике. Затем появились компьютерные интерпретации метода, в результате он стал более надежным, но все же уступал другим методам.

Преимущества:

* Не требуется дорогостоящее оборудование
* Возможно распознание на значительных расстояниях

Недостатки:

* Низкая статистическая достоверность
* Необходимо правильное освещение, например невозможно распознать лицо человека, входящего в помещение с улицы в солнечный день
* Для некоторых систем помехи, вроде очков или бороды, могут быть неприемлемыми
* Некоторые системы не учитывают изменение мимики

### 1.3.4 Распознание по венам рук

Это достаточно новая технология в сфере биометрической идентификации, появилась всего 5-10 лет назад. Инфракрасная камера делает снимки внешней или внутренней стороны руки. Рисунок вен формируется благодаря тому, что гемоглобин крови поглощает ИК излучение. В результате, степень отражения уменьшается, и вены видны на камере в виде черных линий. Специальная программа на основе полученных данных создает цифровую свертку. Не требуется контакта человека со сканирующим устройством.

Преимущества:

* Отсутствие необходимости контактировать со сканирующим устройством
* Высокая достоверность
* Скрытость характеристики, затруднительно получить ее от человека просто так

Недостатки:

* Недопустима засветка сканера солнечными лучами или лучами галогеновых ламп
* Некоторые возрастные заболевания, например артрит, сильно ухудшают достоверность

## 1.4 Идентификация по клавиатурному почерку

Набирая тексты на клавиатуре ежедневно, человек формирует собственный клавиатурный почерк

Клавиатурный почерк - поведенческая биометрическая характеристика, которую описывают следующие параметры:

* Скорость ввода - количество введенных символов разделенное на время печатания
* Динамика ввода - характеризуется временем между нажатиями клавиш и временем их удержания
* Частота возникновение ошибок при вводе
* Использование клавиш - например, какие функциональные клавиши нажимаются для ввода заглавных букв

Человека можно отличить по ряду особенностей:

* зависимость почерка от сочетаний букв в слове
* глубокие связи между набором разных символов
* задержки — при вводе символов

Эти уникальные характеристики выявляются с помощью:

* по набору собственного текста
* по набору ключевой фразы

Механизм идентификации по клавиатурному вводу:

Идентификация по клавиатурному вводу основывается на статистических данных. То есть необходимо сформировать закономерности, например, какие буквы он нажимает быстрее всего, между какими клавишами он меньше всего задумывается и т.д. После этого следует обучение программы, которая будет не только по кодовой фразе идентифицировать человека, но и по его биометрическим особенностям.

Преимущества:

* Простота реализации и внедрения. Реализация исключительно программная, ввод осуществляется со стандартного устройства ввода (клавиатуры), а значит использование не требуется приобретение никакого дополнительного оборудования. Это самый дешевый способ аутентификации по биометрическим характеристикам субъекта доступа.
* Не требует от пользователя никаких дополнительных действий, кроме привычных. Пользователь так или иначе, наверняка, использует пароль, который можно назначить парольной фразой, по которой будет проводиться аутентификация.
* Возможность скрытой идентификации

Недостатки:

* Требуется обучение программы
* Сильная зависимость от эргономичности клавиатуры (в случае смены, придется обучать программу заново)
* Сильная зависимость от психофизического состояния оператора.

# Вторая глава

Для своего исследования я выбрала идентификацию по клавиатурному почерку. Я решила проверить, можно ли опознать человека, по времени задержки между нажатиями клавиш. В ходе работы над этим исследованием, я написала 5 программ, которые поочередно обрабатывают данные.

## Программа для регистрации нажатий клавиш (см. Прил. 5)

В среде программирования OpenProcessing существуют две встроенные функции, которыми я пользовалась для регистрации нажатий: KeyPressed() и KeyReleased().

Первая функция запускается, как только клавиша нажата, а вторая, когда ее отпускают. Следовательно, в функции KeyPressed() записывается время нажатия в миллисекундах с момента начала программы, после чего, в таблицу помещается буква и его время. Анилогично, со временем отпускания. В итоге, сохраняется таблице в файле формата csv. Примером результата данной программы является Таблица 1.

## Программа для выделения смежных букв (см. Прил. 7)

Эта программа предназначена для того, чтобы находить длительность пауз между нажатиями клавиш.

Для начала в таблицу загружается таблица, которая получилась в результате работы предыдущего кода. Программа с помощью цикла с параметром находит разницу между временем нажатия текущей клавиши и временем отпускания предыдущей. И в таблицу записывается буквосочетание длительность паузы между этими символами, после чего она сохраняется в файле формата csv. Примером такой таблицы является Таблица 2.

## Программа для сортировки буквосочетаний(см. Прил. 6)

В результате работы предыдущей программы, мы получаем таблицу с буквосочетаниями в порядке нажатия, для того, чтобы их отсортировать и нужен данный код. С помощью двух циклов, встроенных друг в друга, перебирается таблица. То есть сначала берется первое буквосочетание и затем программа проходит по всей таблице и ищет такие же, после чего записывает их уже новую таблицу в отсортированном виде, блоками. Примером является Таблица 3 (отрывок).

|  |  |
| --- | --- |
| keys | time |
| NE | 129 |
| NE | 157 |
| NE | 119 |
| NE | 441 |
| NE | 187 |
| NE | 171 |
| NE | 283 |
| NE | 146 |
| EV | 111 |
| EV | 101 |
| EV | 59 |
| EV | 195 |
| EV | 136 |
| EV | 167 |
| EV | 73 |
| EV | 138 |
| VE | 110 |
| VE | 79 |
| VE | 72 |
| VE | 114 |
| VE | 114 |
| VE | 148 |
| VE | 105 |

## Программа для нахождения медиан (см. Прил. 8)

Перед тем как писать следующую программу, я задумалась, какие же все-таки паузы брать и почему. Я пришла к выводу, что лучше найти медиану данных значений, так как среднее арифметическое не учитывает разнообразие величин и соотношение абсолютно разных по значению позиций.

Медиана в [математической статистике](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) — число, характеризующее выборку. Если все элементы выборки различны, то медиана — это такое число выборки, что ровно половина из элементов выборки больше него, а другая половина меньше него. В общем виде ее можно найти так. Для этого нужно упорядочить все числа по возрастанию или убыванию и найти средний элемент, если в ряду нечетное количество чисел, Если же нет, то медианой считается полсумма двух средних элементов.

Итак, программа с помощью цикла перебирает таблицу по блокам и находит медиану для каждого буквосочетания. Результатом является Таблица 4.

|  |  |
| --- | --- |
| keys | median |
| NE | 161 |
| EV | 426 |
| VR | 185 |
| ER | 183 |
| R\_ | 126 |
| \_S | 375 |
| SE | 238 |
| EE | 135 |
| EN | 183 |
| N\_ | 140 |
| \_B | 820 |
| BE | 220 |

## apple_hdstore:Users:sonya.akinina:Downloads:chart (1).pngПрограмма для выделения почерка(см. Прил. 9)

Во время выделения почерка возник вопрос: Какие же буквосочетания брать для сравнения? Я решила сравнить гистограммы нескольких таблиц с медианами и вот, что получилось (Рис.1). Для примера я взяла свои данные.

На гистограмме видно, что некоторые паузы совпадают или очень близки, их я и решила брать их как почерк.

Рис.1

В программе я сравниваю три таблицы с медианами, если буквосочетания совпадают и разброс в значениях пауз меньше чем 20%, то я считаю это буквосочетание характерным для данного пользователя и причисляю его к почерку. Результатом работы данного кода является Таблица 5. В приложениях вы можете найти выделенные почерки других пользователей (см. Приложения 1-4).

Таблица 5. Пользователь 1

|  |  |
| --- | --- |
| keys | median |
| VE | 108,875 |
| ER | 139,125 |
| R\_ | 106,75 |
| SE | 157 |
| N\_ | 131 |
| BE | 177,375 |
| EF | 198,5 |
| RE | 157,625 |
| \_N | 158,75 |
| EE | 98 |

После получения почерков нескольких людей, я решила сравнить их и получились вот такие диаграммы (рис.2).



Рис.2

## Программа для проверки подлинности (см. Прил. 10)

Эта программа занимается тем, что выявляет насколько процентов совпала тестовая таблица с почерком. По логике программа мало чем отличается от кода, занимающегося выявлением почерка. В результате работы в консоль выводится процент совпадения.

Для сравнения почерка и тестового ввода я построила гистограмму (Рис.3). На гистограмме видно, что есть буквосочетания, которые почти совпали, по результатам программы это примерно 55 %.



Рис.3

# Заключение

В ходе исследования было диагностировано пять человек, разных возрастов.

Четверо из этой имеют свой почерк и могут быть опознаны алгоритмом с точной от 75%

Так же алгоритм может опознать не того человека, это происходит из-за того, что почерки могут быть похожими или совпадать.

Чтобы улучшить алгоритм, во-первых можно понизить процент разброса. Во-вторых, при формировании почерка и проверки можно вводить минимум слово или предложение (какое-то количество разных символов). Также в уже готовом почерке должно быть какое-то минимальное количество позиций (например, хотя бы половина всех введенных буквосочетаний).

Для увеличения точности составления почерка необходимо увеличить количество вводов. То есть сравнивать большее количество таблиц при формировании почерка пользователя.

При тестовом вводе необходимо увеличить объем текста, что повлияет на точность медиана, а следовательно и на точность сравнения.

Для более индивидуального почерка, можно использовать сочетания из трех букв, почерк будет состоять из сочетаний двух или трех букв, что позволит выделить более четкий почерк.

Из-за того, что каждый раз человек находится в разных состояниях, нужно составить несколько почерком в зависимости, например, от времени суток и сравнивать с ним по ситуации.

# Список литературы

1. Дубровин Д.А «Клавиатурный почерк как способ аутентификации и  идентификации пользователя пользователя» / Дубровин Д.А. Москва: Финансовый университет при правительстве РФ
2. Дубровин Д.А. «Использование клавиатурного почерка для идентификации пользователя» / Дубровин Д.А. Москва: Финансовый университет при правительстве РФ
3. Иванов А.И. Биометрическая идентификация личности по динамике подсознательных движений. – Пенза: ПГУ, 2000.
4. Современные биометрические методы идентификации [Электронный ресурс] / Режим доступа : <https://habrahabr.ru/post/126144/>
5. Применение биометрии [Электронный ресурс] / Режим доступа: <http://www.biometric.bmstu.ru/category/primenenie_biometrii>

# Приложения

***Приложение 1. Пользователь 2***

|  |  |
| --- | --- |
| keys | median |
| NE | 213,625 |
| EV | 93,25 |
| VE | 108,625 |
| ER | 144,375 |
| R\_ | 104,75 |
| SE | 168 |
| EE | 97,5 |
| N\_ | 131 |
| BE | 189,625 |
| EF | 206,5 |
| RE | 166,875 |
| \_N | 164,75 |

***Приложение 2. Пользователь 3***

|  |  |
| --- | --- |
| keys | median |
| ER | 176 |
| R\_ | 124,75 |
| N\_ | 140,625 |
| RE | 158,125 |

***Приложение 3. Пользователь 4***

|  |  |
| --- | --- |
| keys | median |
| \_S | 171,5 |
| OR | 104 |
| FO | 144,625 |
| EN | 180,625 |
| BE | 176,25 |

***Приложение 4. Пользователь 5***

|  |  |
| --- | --- |
| keys | median |
| LO | 140,25 |
| ND | 351,25 |
| DO | 100,75 |
| N\_ | 169,5 |
| S\_ | 101,75 |
| \_A | 180,25 |
| A\_ | 106,25 |
| AP | 206,75 |
| IT | 319 |
| TA | 310,25 |
| L\_ | 186,25 |
| F\_ | 110 |
| \_G | 208 |
| GR | 181,5 |
| RE | 131,25 |
| AT | 111,5 |
| T\_ | 147,5 |
| IN | 221 |

***Приложение 5. Программа для регистрации нажатий клавиш***

var arrayT;

var keyR;

var rows;

var keyP=0;

var timeP=0;

function setup() {

 keyR = 0;

 createCanvas(windowWidth, windowHeight);

 background(100);

 arrayT= new p5.Table(rows);

 arrayT.addColumn('key');

 arrayT.addColumn('timeP');

 arrayT.addColumn('timeR');

}

function keyPressed() {

 if (key==' ') {

 keyP='\_';

 }

 else {

 keyP=key;

 }

 timeP=int(millis());

 println (keyR);

 println(key);

 println ('key pressed');

}

function keyReleased() {

 arrayT.addRow();

 arrayT.setNum(keyR,'timeR',int(millis()));

 arrayT.setString(keyR,'key',key);

 arrayT.setNum(keyR,'timeP',timeP);

 arrayT.setString(keyR,'key',keyP);

 println(key);

 println ('key released');

 keyR=keyR+1;

}

function keyTyped() {

 if (keyCode == ENTER) {

 saveTable(arrayT,'keys.csv');

 print('saving');

 }

}

***Приложение 6. Программа для выделения смежных буквосочетаний***

var arrayT;

var arrayD;

var rowArrayT;

var rowArrayD = 0;

var rowI = 0;

var lettersP1;

var lettersP2;

var lettersP;

var timeP;

var timeR;

var timeD;

var rowCount;

function preload() {

 arrayT = new p5.Table();

 arrayT = loadTable("keys\_18.csv","csv", "header");

}

function setup() {

 createCanvas(windowWidth, windowHeight);

 background(100);

 arrayD = new p5.Table(); //table with processed data

 arrayD.addColumn('keys');

 arrayD.addColumn('timeD');

}

function draw() {

 rowCount = arrayT.getRowCount();

 if (rowI < rowCount) {

 rowArrayT = arrayT.getRow(rowI);

 lettersP1 = rowArrayT.getString("key");

 timeR = rowArrayT.getNum("timeR");

 rowI = rowI + 1;

 rowArrayT = arrayT.getRow(rowI);

 lettersP2 = rowArrayT.getString("key");

 timeP = rowArrayT.getNum("timeP");

 timeD = timeP - timeR; // нахождение промежутка времени между нажатиями

 lettersP = lettersP1 + lettersP2;

 print(lettersP);

 print(timeD);

 arrayD.addRow(); //заполнение таблицы результатов

 arrayD.setString(rowArrayD,"keys",lettersP);

 arrayD.setNum(rowArrayD,"timeD",timeD);

 rowArrayD = rowArrayD + 1;

 }

 if ((rowI == rowCount) || (rowI > rowCount)) {

 saveTable(arrayD,'data.csv');

 print('saving');

 }

}

***Приложение 7. Программа для сортировки данных***

var keyS;

var rowArrayT1;

var rowArrayT2;

var rowI;

var arrayS;

var arrayT;

var rowArrayS = 0;

var rowI = 0;

var lettersS1;

var lettersS2;

var lettersS;

var timeS;

var matchesFound = 0;

var rowCount;

var nan = "\*\*\*";

function preload() {

//грузим таблицу данных (initial - начальная)

 initialTable = new p5.Table();

 initialTable = loadTable("data\_77.csv","csv", "header");

 println ("... table loaded");

}

function setup() {

 createCanvas(windowWidth, windowHeight);

 background(100);

 //создаем новую таблицу, где будем хранить отсортированные данные

 newTable = new p5.Table();

 newTable.addColumn('keys');

 newTable.addColumn('time');

 initialTableRowCount = initialTable.getRowCount();

 print("... found " + initialTableRowCount + " rows");

 noLoop();

}

function draw() {

 var rowToInsertTo = 0;

 initialTableRowCount = initialTable.getRowCount();

 for (rowToInsertTo = 0; rowToInsertTo < initialTableRowCount; rowToInsertTo ++) {

 matchesFound = 0;

 var firstRowInitialTable = initialTable.getRow(rowToInsertTo);

 var phraseToFind = firstRowInitialTable.getString("keys"); //мы выяснили что мы ищем

 var correspondingTime = firstRowInitialTable.getNum("timeD");

 if (phraseToFind != nan) {

 var rowToInsert = newTable.addRow();

 rowToInsert.setString("keys",phraseToFind);

 rowToInsert.setNum("time",correspondingTime);

 println ("... looking for " + phraseToFind + " phrase")

 }

 for (var i = rowToInsertTo+1; i < initialTableRowCount; i ++) {

 var currentRow = initialTable.getRow(i);

 var currentPhrase = currentRow.getString("keys");

 var currentTime = currentRow.getNum("timeD");

 if (phraseToFind == currentPhrase) {

 if (currentPhrase != nan) {

 println (phraseToFind + " matches " + currentPhrase);

 var rowToInsert = newTable.addRow();

 rowToInsert.setString("keys",currentPhrase);

 rowToInsert.setNum("time",currentTime);

 //print(nan);

 initialTable.set(i,"keys","\*\*\*");

 matchesFound ++;

 }

 }

 }

 println (matchesFound + " matches found");

 }

 saveTable(newTable,'data2.csv');

}

***Приложение 8. Программа для нахождение медиан***

var arrayToFindMedian = [];

var NumOfNumbers = 0;

var initialTableRowCount;

var firstRowInitialTable;

var phraseToCheck;

var timeToCheck;

var currentRowInitialTable;

var currentPhraseToCheck;

var correspondingTime;

var rowToInsertTo = 0;

var rowToCheck;

function preload() {

//грузим таблицу данных (initial - начальная)

 initialTable = new p5.Table();

 initialTable = loadTable("data2\_45.csv","csv", "header");

 println ("... table loaded");

}

function setup() {

 createCanvas(windowWidth, windowHeight);

 background(100);

 //создаю новую таблицу

 newTable = new p5.Table();

 print("table created");

 newTable.addColumn('keys');

 newTable.addColumn('median');

 initialTableRowCount = initialTable.getRowCount();

 print("... found " + initialTableRowCount + " rows");

 noLoop();

}

function draw() {

//записываю первое буквосочетание

while (rowToInsertTo < initialTableRowCount) {

firstRowInitialTable = initialTable.getRow(rowToInsertTo);

phraseToCheck = firstRowInitialTable.getString("keys");

correspondingTime = firstRowInitialTable.getNum("time");

println("current row > " + rowToInsertTo);

println("phraseToCheck > " + phraseToCheck);

println("correspondingTime > " + correspondingTime);

//текущая фраза это то, что мы ищем

currentPhraseToCheck = phraseToCheck;

while (phraseToCheck == currentPhraseToCheck) {

 arrayToFindMedian.push (correspondingTime);

 //берем следующую строку

 rowToInsertTo = rowToInsertTo + 1;

 println("row added: current row > " + rowToInsertTo);

 if (rowToInsertTo == initialTableRowCount) {

 break;

 }

 currentRowInitialTable = initialTable.getRow(rowToInsertTo);

 println(currentRowInitialTable);

 currentPhraseToCheck = currentRowInitialTable.getString("keys");

 //println("no problems");

 correspondingTime = currentRowInitialTable.getNum("time");

 println("next currentPhraseToCheck > " + currentPhraseToCheck);

 println("while round completed");

}

 println("arrayToFindMedian complete");

 print (arrayToFindMedian);

 println("mediana calculated");

 median = mediana (arrayToFindMedian);

 print ("mediana > " + median);

 var rowToInsert = newTable.addRow();

 rowToInsert.setString("keys",phraseToCheck);

 rowToInsert.setNum("median",median);

 arrayToFindMedian = [];

 println("arrayToFindMedian emptied");

 println(" ");

 println("\_\_\_\_\_\_\_\_\_\_\_\_\_block complete\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

 saveTable(newTable,'medians.csv');

 println ("saving");

}

function mediana (array) {

var median = 0;

var array = sort(array,array.length);

if (array.length % 2 == 0) {

 median = (array[array.length/2-1]+array[array.length/2])/2;

}

else {

 median = array[floor(array.length/2)]; }

return (median);

}

***Приложение 9. Программа для выделения почерка***

var handWritingID = 0;

var percent = 0;

var countPhrases = 0;

var countMatches = 0;

var handWritingTableRowCount = 0;

var testTableRowCount = 0;

function preload() {

handWritingTable = new p5.Table();

handWritingTable = loadTable("medians\_32.csv","csv", "header");

println ("... table1 loaded");

testTable = new p5.Table();

testTable = loadTable("handwriting\_20.csv","csv", "header");

println ("... table2 loaded");

}

function setup() {

createCanvas(windowWidth, windowHeight);

background(100);

handWritingTableRowCount = handWritingTable.getRowCount();

testTableRowCount = testTable.getRowCount();

println (handWritingTableRowCount + " in the first table");

println (testTableRowCount + " in the first table");

newTable = new p5.Table();

print("table created");

newTable.addColumn('keys');

newTable.addColumn('median');

noLoop();

}

function draw() {

for (var i = 0; i < handWritingTableRowCount; i++) {

var RowHandWritingTable = handWritingTable.getRow(i);

var phraseToCheck = RowHandWritingTable.getString("keys");

var medianToCheck = RowHandWritingTable.getNum("median");

 for (var j = 0; j < testTableRowCount; j++) {

 var RowTestTable = testTable.getRow(j);

 var currentPhrase = RowTestTable.getString("keys");

 var currentMedian = RowTestTable.getNum("median");

 if (phraseToCheck == currentPhrase) {

 println (phraseToCheck + " mathces " + currentPhrase);

 countPhrases++;

 if ((currentMedian >= medianToCheck - medianToCheck\*0.2) && (currentMedian <= medianToCheck + medianToCheck\*0.2)) {

 println ("match found");

 var median = (currentMedian + medianToCheck)/2;

 var rowToInsert = newTable.addRow();

 rowToInsert.setString("keys",phraseToCheck);

 rowToInsert.setNum("median",median);

 countMatches++;

 }

 }

 }

}

println (countPhrases + " same letter(s) analyzed " + countMatches + " match(es) found " + 100\*countMatches/countPhrases + " similarity")

saveTable(newTable,'handwriting.csv');

}

***Приложение 10. Программа для проверки на подлинность***

var handWritingID = 0;

var percent = 0;

var countPhrases = 0;

var countMatches = 0;

var handWritingTableRowCount = 0;

var testTableRowCount = 0;

function preload() {

handWritingTable = new p5.Table();

handWritingTable = loadTable("handwriting\_11.csv","csv", "header");

println ("... table1 loaded");

testTable = new p5.Table();

testTable = loadTable("medians\_33.csv","csv", "header");

println ("... table2 loaded");

}

function setup() {

createCanvas(windowWidth, windowHeight);

background(100);

handWritingTableRowCount = handWritingTable.getRowCount();

testTableRowCount = testTable.getRowCount();

println (handWritingTableRowCount + " in the first table");

println (testTableRowCount + " in the first table");

noLoop();

}

function draw() {

for (var i = 0; i < handWritingTableRowCount; i++) {

var RowHandWritingTable = handWritingTable.getRow(i);

var phraseToCheck = RowHandWritingTable.getString("keys");

var medianToCheck = RowHandWritingTable.getNum("median");

 for (var j = 0; j < testTableRowCount; j++) {

 var RowTestTable = testTable.getRow(j);

 var currentPhrase = RowTestTable.getString("keys");

 var currentMedian = RowTestTable.getNum("median");

 if (phraseToCheck == currentPhrase) {

 println (phraseToCheck + " mathces " + currentPhrase);

 countPhrases++;

 if ((currentMedian >= medianToCheck - medianToCheck\*0.4) && (currentMedian <= medianToCheck + medianToCheck\*0.4)) {

 println ("match found");

 countMatches++;

 }

 //else {countMatches = countMatches - 1; }

 }

 }

}

println (countPhrases + " same letter(s) analyzed " + countMatches + " match(es) found " + 100\*countMatches/countPhrases + " similarity")

}