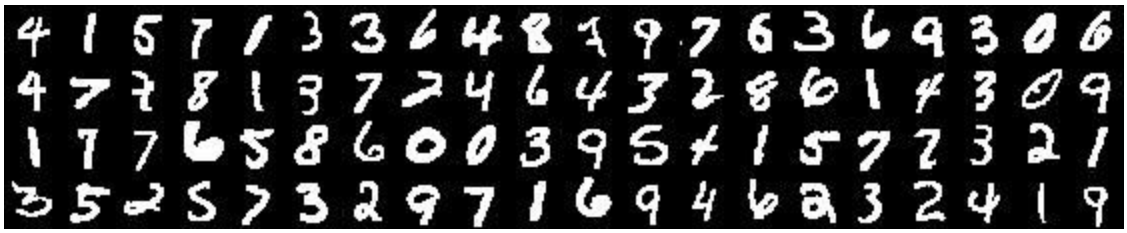
# Задача

Реализовать программу для классификации рукописных цифр.

# Описание

В качестве исследуемого набора данных будет использоваться база изображений рукописных цифр MNIST (см. рисунок). Изображения в данной базе имеют разрешение 28x28 и хранятся в виде набора значений оттенков серого. Вся база разбита на две части: тренировочную, состоящую из 50000 изображений, и тестовую — 10000 изображений.



Создатели базы данных NIST  использовали набор образцов из Бюро переписи населения США, к которому были добавлены ещё тестовые образцы, написанные студентами американских университетов.

Для решения этой задачи предлагается применить **Метод k ближайших соседей**[1] — метрический алгоритм для автоматической классификации объектов. Основным принципом метода ближайших соседей является то, что объект присваивается тому классу, который является наиболее распространённым среди соседей данного элемента. Соседи берутся исходя из множества объектов, классы которых уже известны, и, исходя из ключевого для данного метода значения k рассчитывается, какой класс наиболее многочислен среди них. Каждый объект имеет конечное количество атрибутов (размерностей). В качестве расстояния между объектами можно использовать **Евклидову метрику,** то есть привычное нам расстояние между точками в пространстве.

[1] ru.wikipedia.org/wiki/Метод_k-ближайших_соседей

# Требования

Необходимо спроектировать и написать класс (или несколько классов), который будет реализовывать алгоритм распознавания. Должна быть возможность инициализировать класс данными для обучения и предоставить метод для распознавания одного изображения.

Кроме самой реализации алгоритма, следует написать код для проверки его точности. Для этого следует использовать 10000 тестовых данных — распознать 10000 заранее известных изображений и вычислить количество ошибок. При этом, нужно уметь разбивать 10000 изображений на равные части и вычислять точность в несколько потоков.

Кроме вычисления точности предлагается провести эксперимент: вместо Евклидовой метрики использовать Расстояние_городских_кварталов или даже угол между векторами и проверить качество распознавания.

# Выходные данные

1. В процессе работы программа должна выводить количество обработанных тестовых данных в процентах (прогресс) в виде:

   ```
   Завершено: 55%
   ```

   Для этого можно использовать возврат каретки (\r).

2. После завершения работы программа выводит:

   ```
   Завершено: 100%
   Количество ошибок: …
   Точность распознавания: …%
   ```

# Дополнительно

## Внешние библиотеки

Использование внешних библиотек не требуется, однако и не запрещено (за исключением самой реализации алгоритма).

Например, можно использовать Apache Commons IO, с помощью которой можно упростить чтений файлов:

```
byte[] images = IOUtils.toByteArray(new FileInputStream(new File(fileImages)))
```

## Знаковые байты

В процессе разработки обнаружится, что значения цветов в исходных данных — это беззнаковые байты (0-255), в то время как в Java байт знаковый (от -128 до 127).

Можно беззнаковые байты перевести в знаковый int:

```
int a = b & 0xff
```

# Ссылки

MNIST handwritten digit database
Метод k ближайших соседей — Википедия
Евклидова метрика