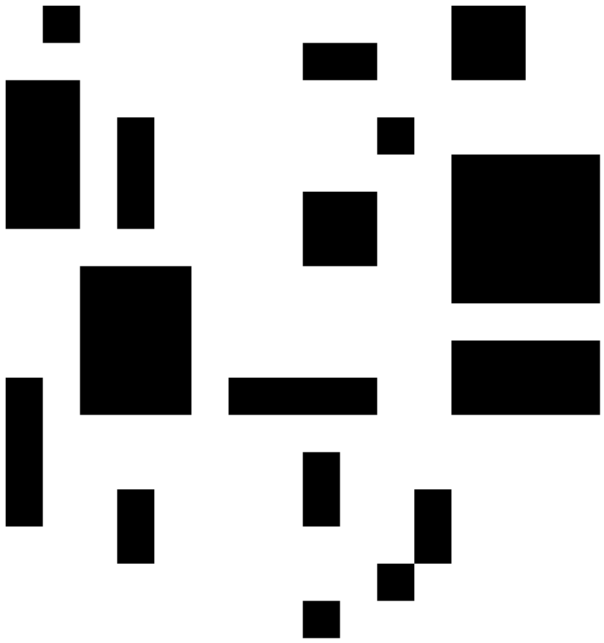
Отчет

Изначально создаётся матрица 20 на 20 элементов, заполняется 1 – это наше пустое пространство, затем добавляются препятствия прямоугольной формы рандомно раскиданные по пространству, но не пересекающиеся. Количество препятствий можно регулировать в коде.



Для построения пути выбираются две рандомные точки (два элемента матрицы = 1) далее формируются список для элементов графа и дистанции и точка старта берётся за нулевую. Также формируется список уже посещённых элементов.

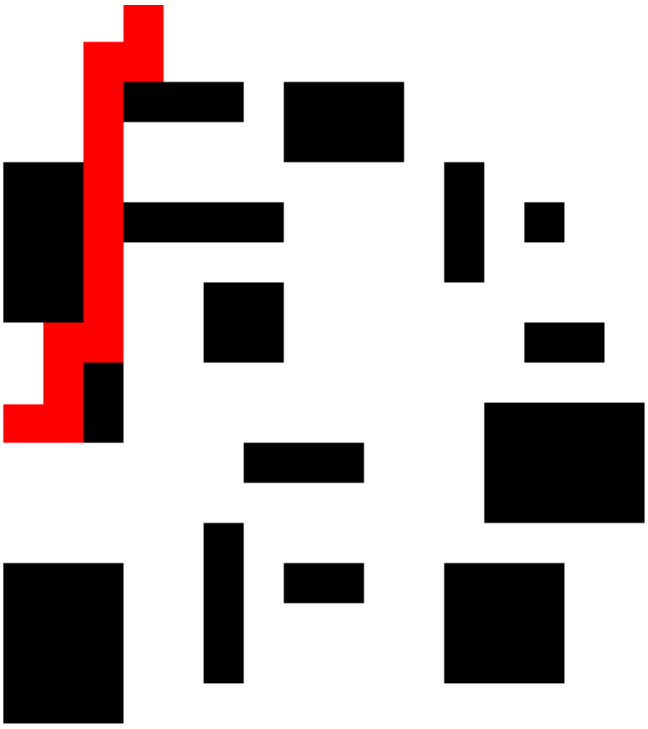
Реализация алгоритма:

выбирается элемент с минимальной дистанцией, проверяется, не является ли он конечным, и записывается в список, затем список инвертируется, чтоб путь был в нужном порядке, далее интеграция по соседним элементам с проверкой на "посещение" и нарушение границ матрицы. После следует выбор нового элемента на основе расчёта дистанции. Каждый пройденный элемент добавляется в список посещённых.

Отрисовка:

Пройденный путь в матрице меняется с 1 на 5.

Матрица переводится в png, где 1 - белые пиксели, 2 - чёрные пиксели (препятствия), 5 - красные пиксели(путь) a*



создаётся функция astar

в ней формируют списке аналогично предыдущей программе.

Далее идёт основной цикл a*

берётся узел с минимальным расстоянием

- проверка на конечный элемент
- добавление в список посещённых
- итерация по соседним элементам с проверкой на нахождение соседей внутри матрицы, а также равны ли они 1 и проверка на "посещение"

Вычисляется новое расстояние, проверяется короче ли он старого и записывается

Если конечный элемент не достижим, функция выводит None.

После описания функции используется тот же код, что и в программе с Дейкстра для создания матрицы и прямоугольных препятствий.

для построения пути используется начальные координаты, конечные и функция, которая записывает все элементы пути в список path

После идёт замена элементов пути на 5 и отрисовка.