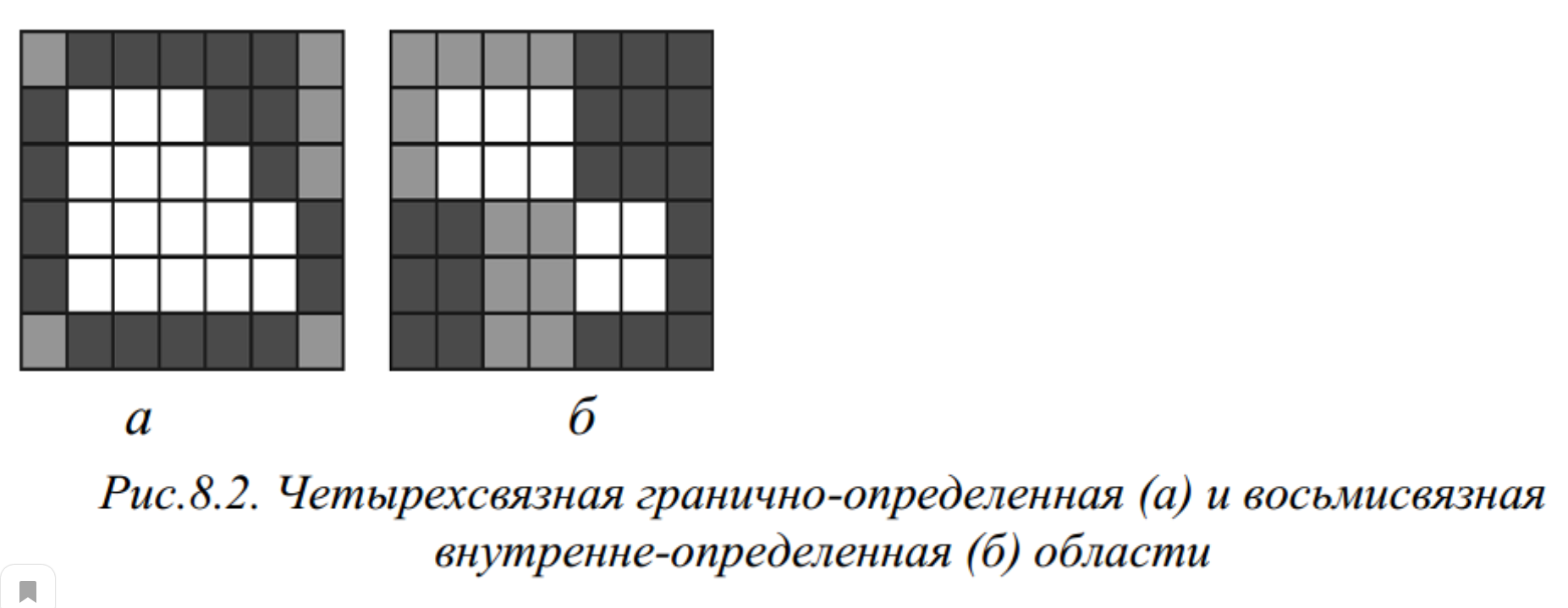
" Программная реализация алгоритмов заполнения с затравкой и исследование их временных характеристик"

Цель работы: реализовать построчный алгоритм заполнения гранично-определенной области с затравкой.

В алгоритмах затравочного заполнения сплошных областей предполагается, что известна определенная точка (затравка) внутри области и необходимо определить точки, соседние с затра­вочной и расположенные внутри области. Если соседняя точка рас­положена не внутри, значит обнаружена граница области, если же точка находится внутри, то она становится новой затравочной точкой и поиск продолжается рекурсивно.

Известны два алгоритма заполнения с затравкой: простой алгоритм заполнения с затравкой; построчный алгоритм заполнения с затравкой. Данные алгоритмы применимы к ***гранично-определенным областям***, то есть таким, что все пиксели на границе данных об­ластей имеют выделенное значение или цвет, но ни один пиксель из внутренней части таких областей не может иметь это выделен­ное значение. (Пиксели, расположенные на границе ***внутренне-определенной*** области, могут иметь разные цвета, за исключением цвета самой области.) Гранично-определенные области могут быть 4- или 8-связными. Любой пиксель в 4-связной области становится доступен с помощью движений только в четырех направ­лениях: вверх, вниз, направо, налево; для 8-связной же области к любому пикселю можно подойти. используя комбинацию движений в 2-х горизонтальных, в 2-х вертикальных и 4-х диагональных нап­равлениях. ***Алгоритм заполнения 8-связной области заполнит и 4-связную область (обратное неверно).***

***Отметим также то, что граница четырехсвязной области является восьмисвязной, а граница восьмисвязной области – четырехсвязной.***

******

Простой алгоритм заполнения с затравкой (базовый алгоритм разработан Смитом, см) легко реализовать, используя по­нятие стека с порядком обслуживания "первым пришел, последним обслужен", то есть, когда новое значение помещается в стек, то все остальные опускаются вниз на один уровень, а когда значение извлекается из стека, все остальные поднимаются на один уро­вень. Такой стек еще называется стеком прямого действия.

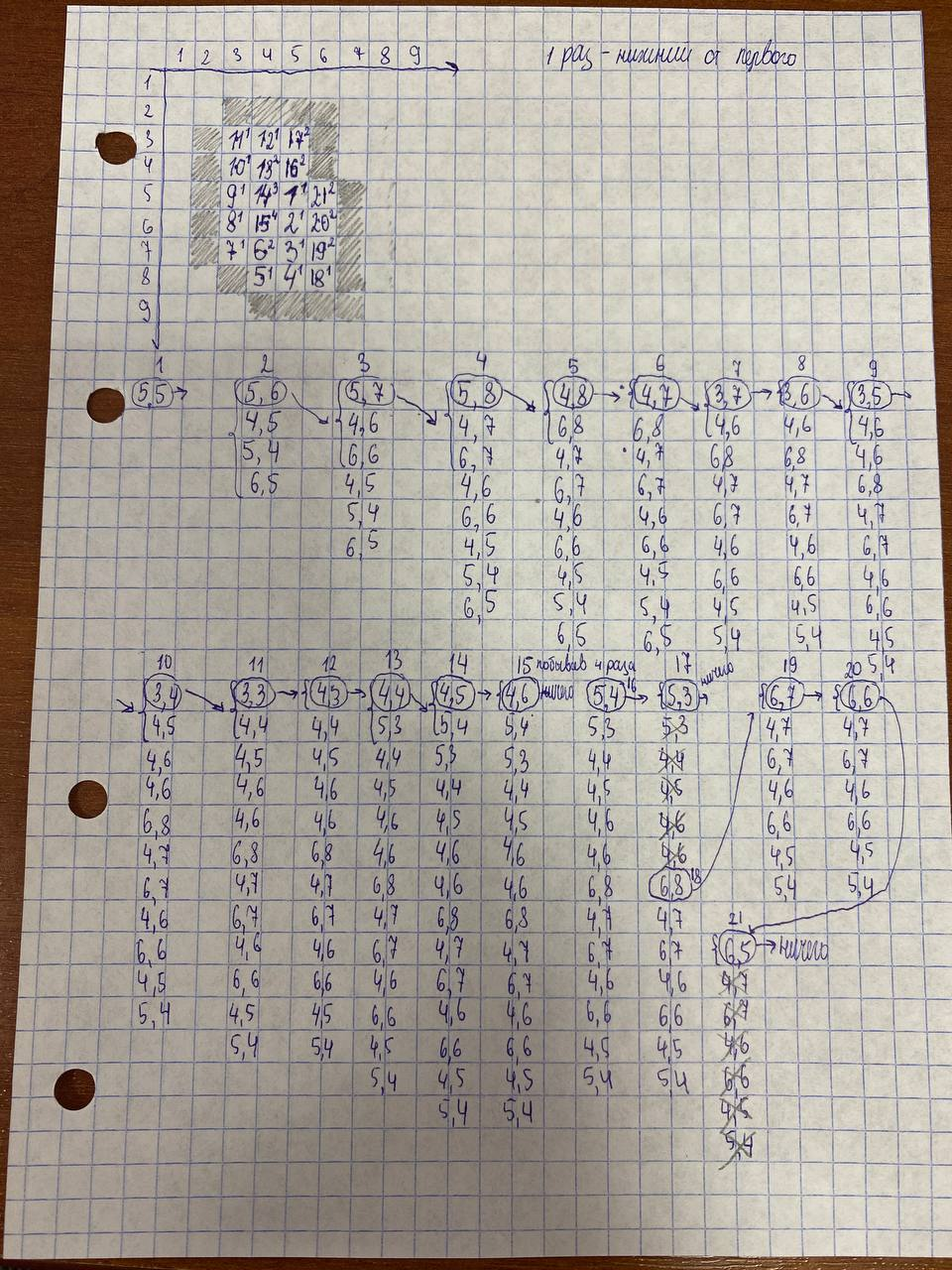
Процесс реализации ***простого алгоритма с затравкой*** для 4-связанной области состоит в следующем:

1. Ввод исходных данных (информация о границах области, цвет границы, цвет заполнения, координаты затравки)
2. Занесение в стек затравочного пиксела. (Затравка выдает затравочный пиксель (x,y), который затем помещается в стек. Стек инициализируется.)
3. Осуществляется проверка на наличие пиксе­лей в стеке. Пока стек не пуст, то:
   1. Затравочный пиксель (x, y) извлекается из стека
   2. Закрасить пиксель (если время закраски > времени получения информации, то можно сначала проанализировать его цвет, и если он не закрашен, закрасить) (получается, если пиксель уже закрашен, то он побывал в стеке, и его соседи уже проанализированы и занесены в стек, их вновь заносить не надо)
   3. Проводится анализ: надо ли помещать соседние пиксели в стек. Для этого, каждый из соседних к данному 4-связанных пик­селя проверяется на два условия: не является ли он граничным или не присвоено ли уже ему требуемое значение:
      1. если (пиксель (x+1, y) < > граничное зн-е и пиксель(x+1,y) < > требуемое зн-е), тогда пиксель (x+1,y) помещается в стек;
      2. если (пиксель (x,y+1) < > граничное зн-е и пиксель(x,y+1) < > требуемое зн-е), тогда пиксель (x,y+1) помещается в стек;
      3. если (пиксель (x-1,y) < > граничное зн-е и пиксель(x-1,y) < > требуемое зн-е), тогда пиксель (x-1,y) помещается в стек;
      4. если (пиксель (x,y-1) < > граничное зн-е и пиксель(x,y-1) < > требуемое зн-е), тогда пиксель (x,y-1) помещается в стек.

То есть, если проверка какого-либо из двух случаев дает положительный результат, то пиксел игнорируется, в противном случае этот пиксель помещается в стек. В приведенном алгоритме 4-связанные ***пиксели проверяются, начиная с правого от текущего, в направлении против часовой стрелки***. Данный алгоритм может за­полнять и области, содержащие дыры.

***«-«***

* все неграничные, не закрашенные пикселы области помещаются в стек – большой расход памяти
* пиксел рассматривается несколько раз и заносится в стек несколько раз (до 4/3?? раз включительно)



Улучшить простой алгоритм можно, если в стек заносить только один пиксель для ***непрерывного интервала пикселей*** (группа примыкающих друг к другу не закрашенных пикселей, ограниченная граничными или закрашенными пикселями).

В заданном интервале в качестве затравочного выбирается ***самый правый*** пиксель из возможных.

В алгоритме ***2 шага***

1. заполнение пиксели текущей строки (влево и вправо от затравочного)

2. поиск новых затравочных пикселей (на двух соседних строках по отношению к текущей) в интервале xл <= x <= xпр

***Расширение границ происходит на 1 этапе***

На лабораторную работу выносится реализация ***построчного алгоритма с затравкой***, поэтому рассмотрим подробнее построчный алгоритм заполнения с затравкой для 4-связной области (для са­мостоятельной работы предлагается переделать его для 8-связан­ной области и заполнение проводить не в 4-х, а в 8-ми направ­лениях). Гранично-определенная 4-связанная область может быть выпуклой, не выпуклой, а также содержать внутри себя дыры; но ***во внешней, примыкающей к данной гранично-определенной области, не должно быть пикселей с цветом заполнения.***

В алгоритме пункты 1-3.2 остаются

1. Ввод исходных данных (информация о границах области, цвет границы, цвет заполнения, координаты затравки)
2. Занесение в стек затравочного пиксела. (Затравка выдает затравочный пиксель (x,y), который затем помещается в стек. Стек инициализируется.)
3. Осуществляется проверка на наличие пиксе­лей в стеке. Пока стек не пуст, то:
   1. Затравочный пиксель (x, y) на интервале извлекается из стека
   2. Интервал с затравочным пикселем (включая затравочный) заполняется влево и вправо от затравочной точки вдоль сканирующей строки до тех пор, пока не будет найдена граница слева и справа, запоминаем границы:
      1. Заполняем интервал вправо от затравки: x=x+1 (не надо, чтобы заполнить затравочный), пока (цвет(x,y)!=граничное зн-е)делать:(цвет(x,y)=требуе­мое зн-е; x=x+1);
      2. В переменной xr запоминаем крайний правый пиксель: xr=x-1.
      3. Восстанавливаем координату абсцисс затравки: x=xt.
      4. Заполняем интервал слева от затравки: x=x-1, пока (пиксель (x,y)!= граничное зн-е) делать:( пиксель(x,y)=требуе­мое зн-е; x=x-1);
      5. В переменной xl запоминаем крайний левый пиксель: xl= x+1.
      6. Восстанавливаем координату абсциссы затравки: x=xt.(зачем??
   3. Поиск новых затравочных пикселей в интервале xл <= x <= xпр на двух соседних строках y+1, y-1 (поиск первого затравочного (неграничного и незакрашенного) пиксела и в случае его нахождения – поиск самого правого затравочного пиксела и занесение его в стек. Если интервал прерывается граничными или закрашенными пикселями, то необходимо найти новый интервал не закрашенных пикселей и найти в нем самый правый затравочный пискель)

Строка «ниже текущей в экранной системе)

X = xл, y = y+1

В цикле пока x<=xпр ищем затравочный пиксель:

* + 1. Flaf=0 (флаг перехождения затравки)
    2. Пока (цвет (x, y) не граничный) и (цвет (x, y) не заполнения) и (x < xпр), то (если flag=0, то flag=1) и x=x+1
    3. по­мещаем в стек крайний справа пиксель: Если (flag=1), то
       1. Если (цвет (x, y) не граничный) и (цвет (x, y) не заполнения) и (x = xпр), то занесение в стек пиксела (x, y)
       2. Иначе – занесение в стек (x-1, y)
    4. Поиск нового интервала в случае прерывания текущего интервала (если x< xпр) (Далее проверка не является ли строка ниже текущей границей многоугольника, или уже полностью заполненной продолжается в том случае, если интервал был прерван)
       1. xn=x (запоминаем абсциссу текущего пиксела), flag=0
       2. Пока ((цвет (x, y) = граничный) или (цвет (x, y) = заполнения)) и (x<xпр), то x = x+1
       3. убедимся, что координата абсциссы пикселя уве­личилась: Если x=xn, то x=x+1, повторение всех действий для нового полуинтервала
  1. Аналогично пункту 3.3. осуществляем проверку строки ни­же текущей, изменения лишь касаются координаты y, ей присваива­ется значение: y=y-1.

И так до тех пор, пока стек не становит­ся пустым, область заполненной, а работа алгоритма завершен­ной. ***Если гранично-определенная область примыкает к краю экра­на***, то при реализации алгоритмов заполнения с затравкой возмо­жен выход за пределы области, чтобы этого не случилось рекомен­дуется: - либо на каждом шаге производить проверку, не превзой­дены ли пределы памяти; - либо сразу же увеличивать пределы па­мяти на краях, для исключения возможности появления ошибки.