Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 5

Дисциплина Компьютерная графика

Тема Реализация и исследование алгоритмов растрового заполнения сплошных областей

Студент Хетагуров П.К.

Группа ИУ7-45

Оценка (баллы) _____

Преподаватель Куров А. В

Москва.
2020 г.

Вариант 26. Алгоритм заполнения по ребрам

Цель работы:

Реализация и исследование одного из алгоритмов (по заданию преподавателя) растрового заполнения области.

Задание:

В рамках данной работы необходимо реализовать один из алгоритмов заполнения сплошной области.
Необходимо обеспечить ввод произвольной многоугольной области, содержащей произвольное количество отверстий.
Ввод (вершин многоугольника) производить с помощью мыши, при этом для удобства пользователя должны отображаться ребра, соединяющие вводимые вершины. Предусмотреть ввод горизонтальных и вертикальных ребер.
Пользователь должен иметь возможность задания цвета заполнения.
Работа программы должна предусматривать два режима – с задержкой и без задержки.
Режим с задержкой должен позволить проследить выполняемую последовательность действий.
(Задержку целесообразно выполнять после обработки очередной строки).
Обеспечить замер времени выполнения алгоритма (без задержки, с выводом на экран только окончательного результата).

Теоретическая часть:

Суть алгоритма:

Если цвет заполнения является инверсией цвета фона и наоборот, то для каждой сканирующей строки, пересекающей ребро многоугольника дополнить все пиксели, расположенные правее точки пересечения.

Можно ограничить рассматриваемые пиксели максимальной абсциссой из абсцисс вершин. Т.е вводится перегородка. Но, в отличие от алгоритма с перегородкой, перегородка всегда проходит через вершину с максимальной абсциссой. Это существенно уменьшит кол-во пикселей, обрабатываемых зря (они точно находятся вне многоугольника).

Преимущества алгоритма:

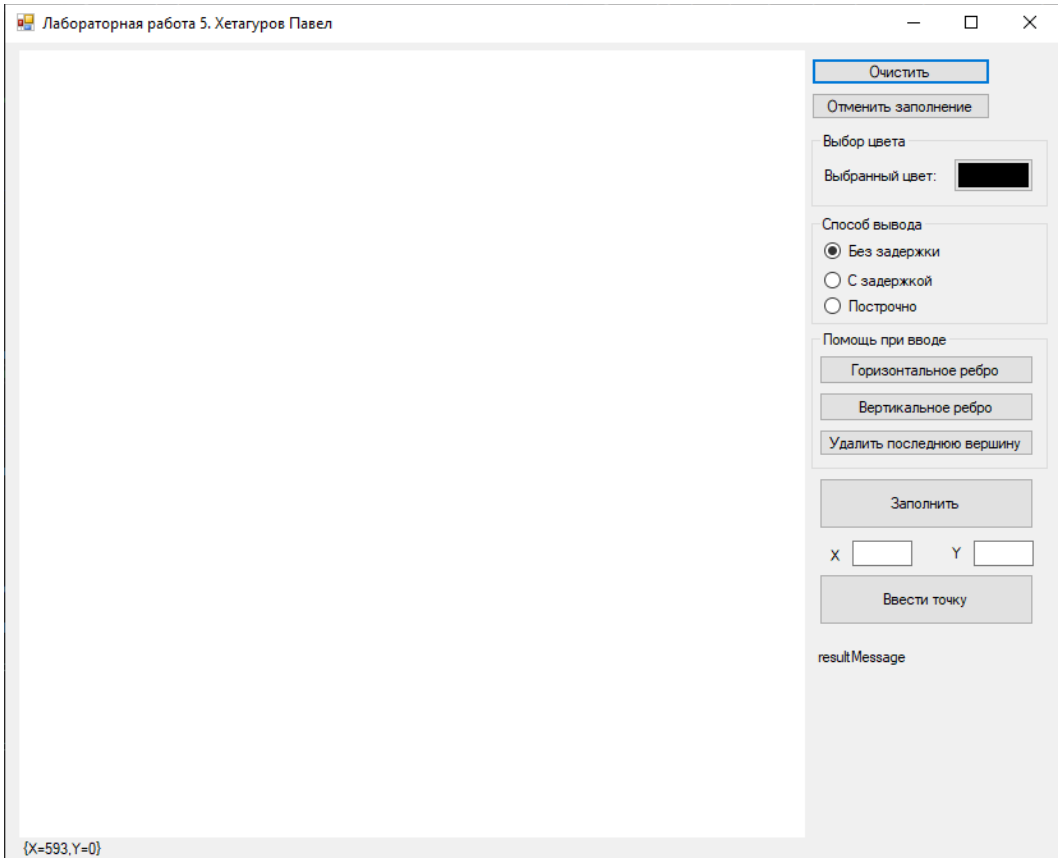
1) Алгоритм заполнения по ребрам не требует создание и сортировки списка ребер (в отличие от алгоритма с упорядоченными ребрами).
2) Алгоритм применяется к каждому ребру индивидуально и может обрабатывать ребра в произвольном порядке.
3) Простота

Недостатки алгоритма:

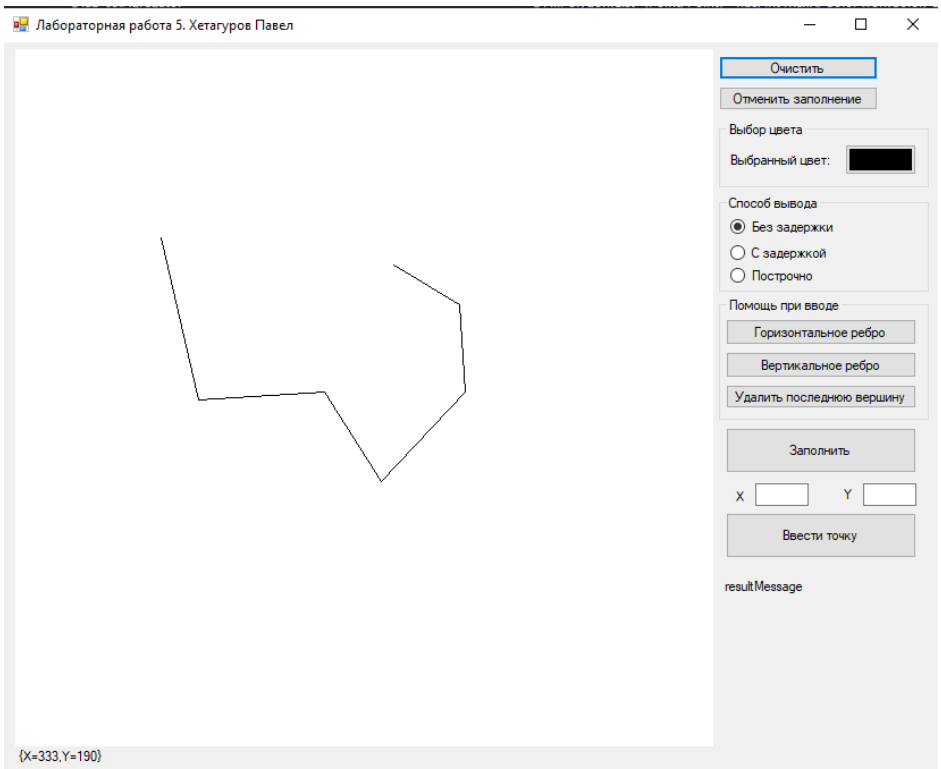
1) Один и тот же пиксель обрабатывается большое количество раз (столько, сколько ребер сканирующая строка, на которой он лежит, пересекает слева от него)
2) Обрабатываются пикселы вне многоугольника.
3) Так как происходят обращения к цвету пикселя (считывание и изменение), скорость алгоритма также зависит от скорости считывания и изменения цвета пиксела. (Если цвет заполнения является инверсией цвета фона и наоборот, то считывание цвета опускается)

Практическая часть

Интерфейс:

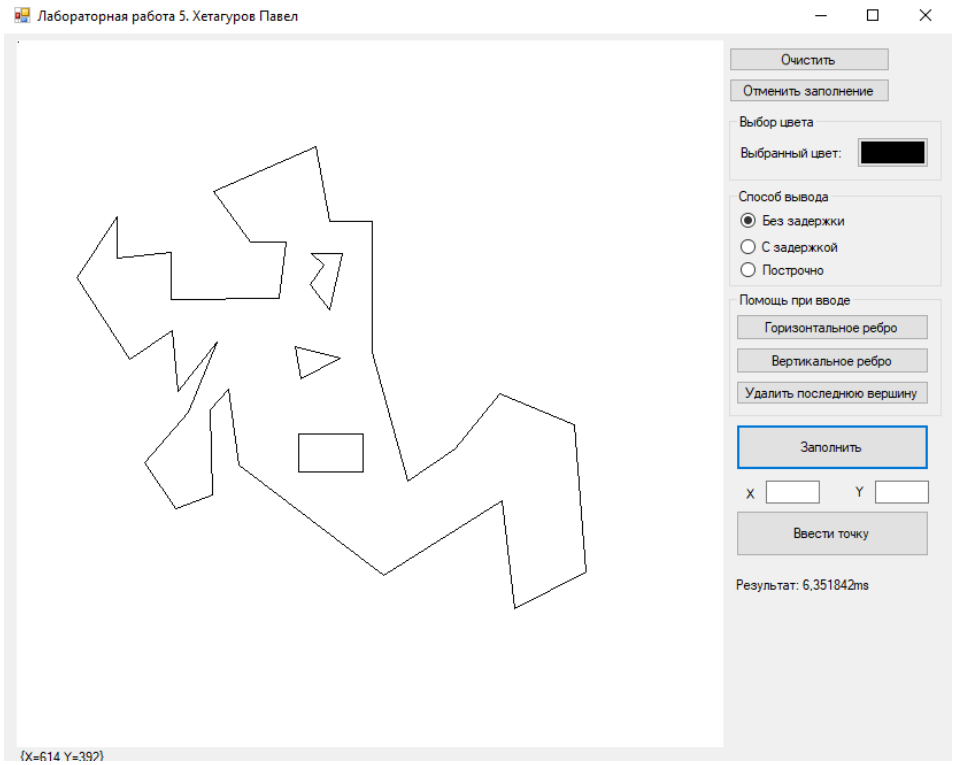


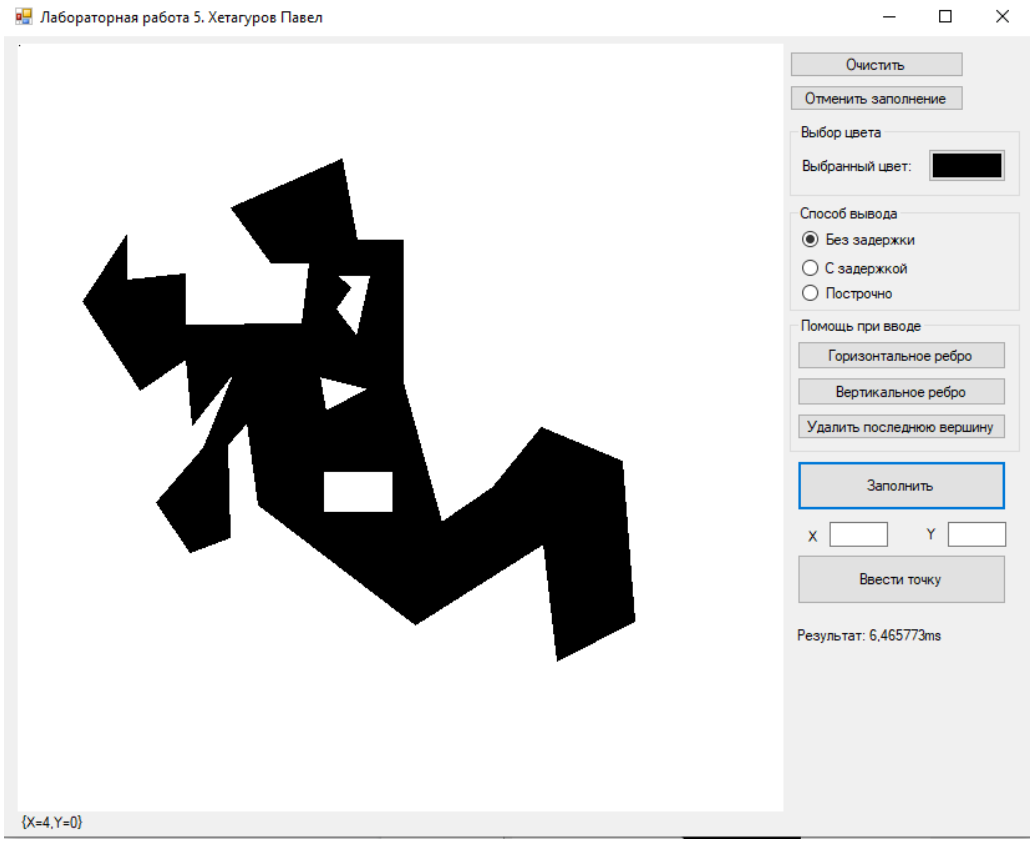
Вводить вершины можно как с помощью мыши (по нажатию на область рисования), так и с помощью текстовых полей (по координатам)



При вводе вершин они соединяются с помощью ребер.

Заполнение произвольной сложной фигуры:

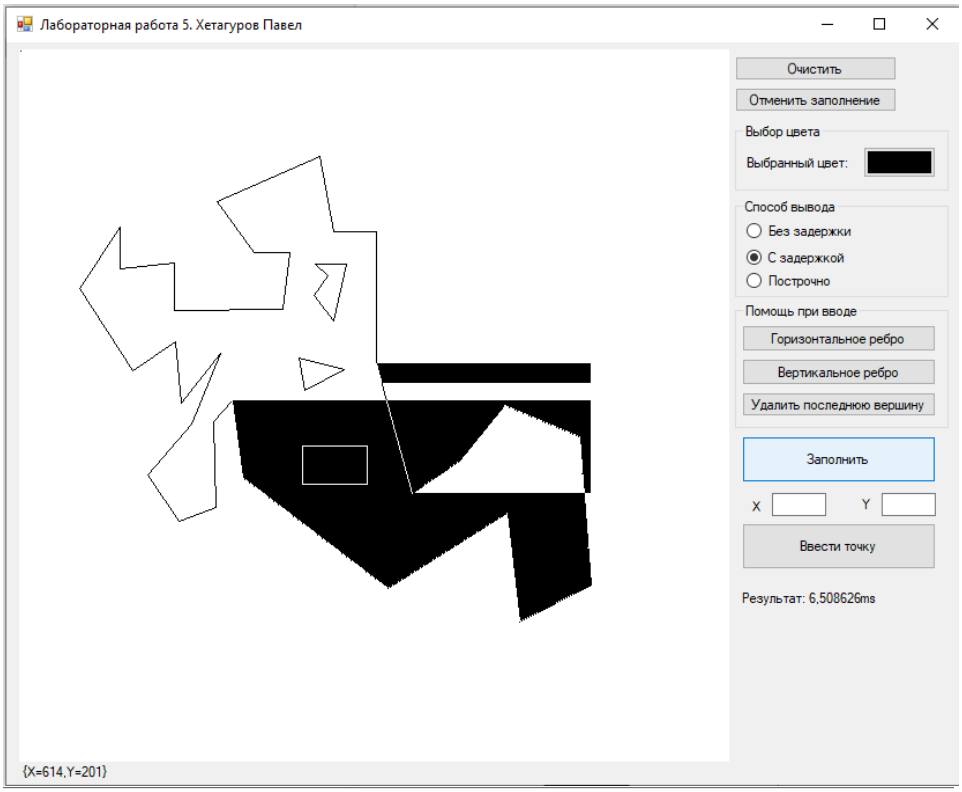


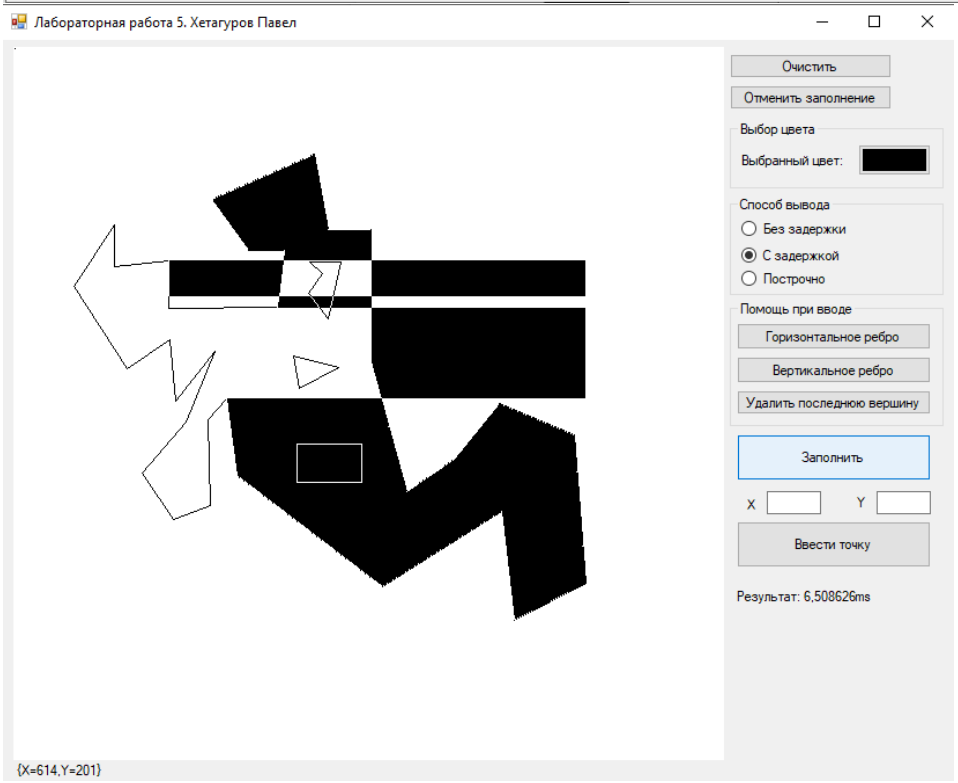


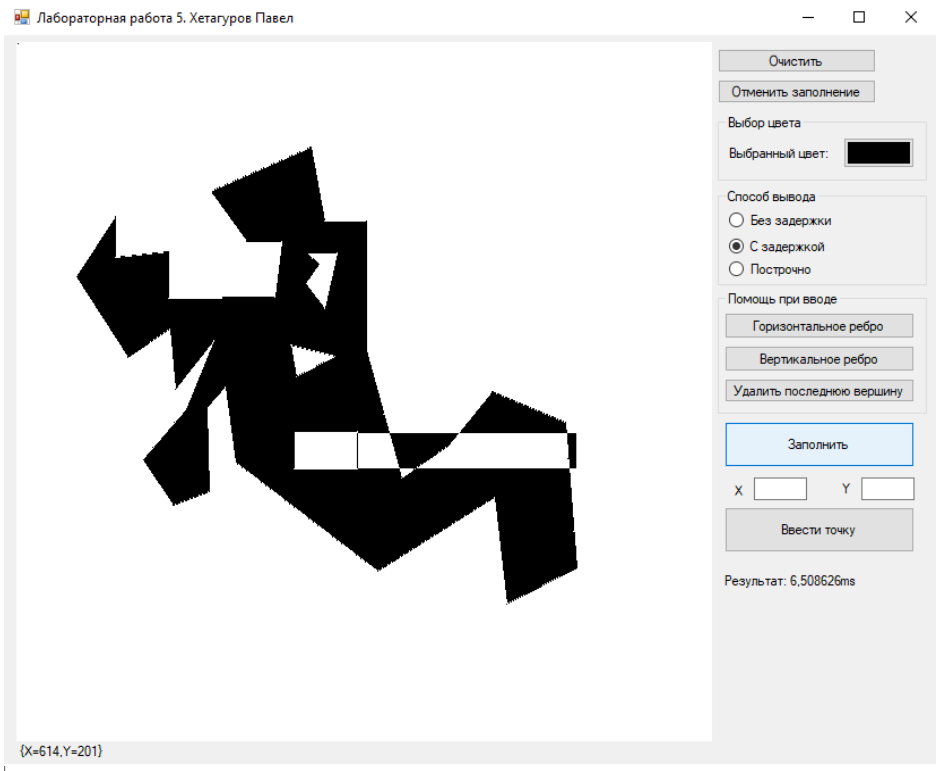
(На скриншоте появляются артефакты на границах фигуры, на экране их нет. Скорее всего из-за сжатия)

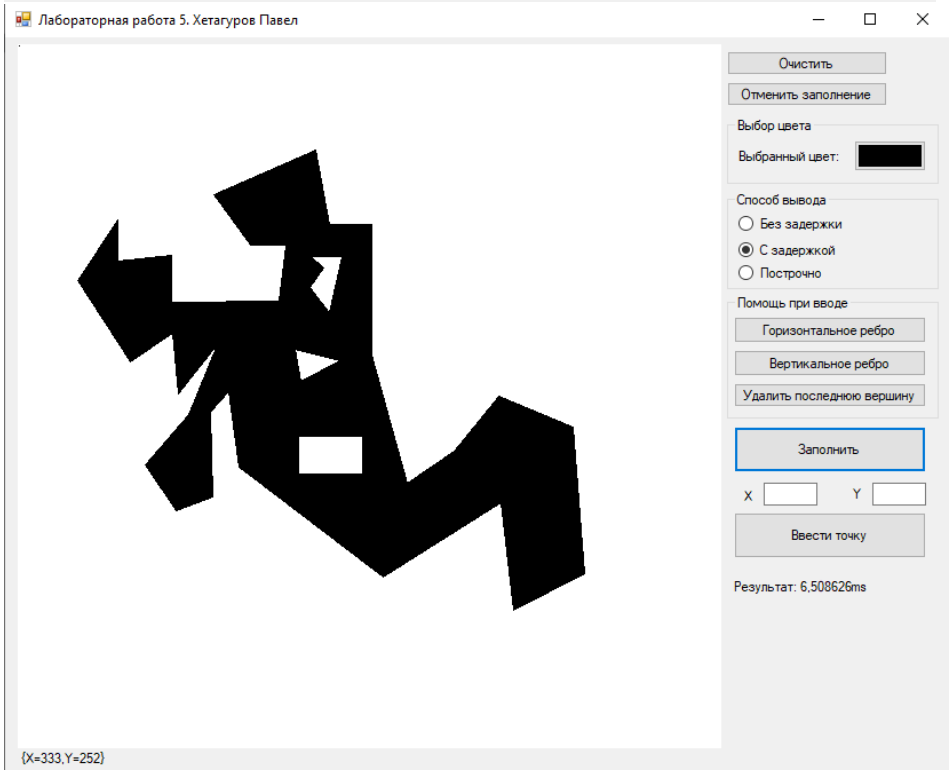
Время полного заполнения: 6.47 миллисекунд

При заполнении с задержкой можно проследить промежуточные результаты:


Лабораторная работа 5. Хетагуров Павел
Очистить
Отменить заполнение
Выбор цвета
Выбранный цвет:
Способ вывода
Без задержки
С задержкой
Построчно
Помощь при вводе
Горизонтальное ребро
Вертикальное ребро
Удалить последнюю вершину
Заполнить
X
Y
Ввести точку
Результат: 6,508626ms
{X=614,Y=201}


Лабораторная работа 5. Хетагуров Павел
Очистить
Отменить заполнение
Выбор цвета
Выбранный цвет:
Способ вывода
Без задержки
С задержкой
Построчно
Помощь при вводе
Горизонтальное ребро
Вертикальное ребро
Удалить последнюю вершину
Заполнить
X
Y
Ввести точку
Результат: 6,508626ms
{X=614,Y=201}





Код:

```
internal static void FillPolygon(List<(Point, Point)> ribs, int maxX,
Color workColor, Color backColor, Bitmap workBitmap)
    {
```

```
        Point first;
        Point second;
        int dY;
        double dX;
        int currentY;
        double currentX;

        foreach (var rib in ribs)
        {
            first = rib.Item1;
            second = rib.Item2;

            // Игнорируем горизонтальные ребра
            if (first.Y == second.Y)
            {
                continue;
            }
            // Для корректной обработки вершин все ребра обрабатываем в
одном направлении
            else if (first.Y > second.Y)
            {
                Point temp = first;
                first = second;
                second = temp;
            }

            dY = second.Y - first.Y;
            dX = (double)(second.X - first.X) / dY;
            currentX = first.X + 1; // начинаем с ближайшего правого
пиксела
            currentY = first.Y;

            for (int i = 0; i < dY; i++)
            {
                for (int x = (int)currentX; x < maxX; x++)
                {
                    // т.к цвет заливки может не быть дополнением
фонового, то надо смотреть на цвет пикселя => лишнее сравнение и долгие
операции считывания/записи

                    Color tempColor = workBitmap.GetPixel(x, currentY);

                    if (tempColor.R == backColor.R && tempColor.G ==
backColor.G && tempColor.B == backColor.B)
                    {
                        workBitmap.SetPixel(x, currentY, workColor);
                    }
                    else
                    {
                        workBitmap.SetPixel(x, currentY, backColor);
                    }
                }
                currentX += dX;
                currentY += 1;
            }
        }

        DrawRibs(workBitmap, ribs, workColor);
```

```
    }
```