|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 № 6**

|  |  |
| --- | --- |
| **Тема \_РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ АЛГОРИТМА ПОСТРОЧНОГО ЗАТРАВОЧНОГО ЗАПОЛНЕНИЯ СПЛОШНЫХ ОБЛАСТЕЙ**  **Студент \_Турсунов Ж. Р.\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-46Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Куров А. В.\_\_\_\_** |  |

Москва.

2020 г.

**Цель работы:** Реализация и исследование алгоритма построчного затравочного заполнения.

**Входные данные:**  Координаты вершин многоугольника, цвет заполнения, выбор режима — с задержкой и без.

**Выходные данные:** Пользовательское меню, содержащее поля ввода и холст с конечным изображением. Вывод замера времени выполнения алгоритма( без задержки ).

**Ошибочные ситуации:** Программа прекращается, если хотя бы один из входных данных не корректен.

**Теоретическая часть**

Алгоритмы заполнения с затравкой основываются на следующем подходе: в них предполагается, что известен хотя бы один пиксель из внутренней области фигуры, отталкиваясь от которого алгоритм находит и закрашивает все остальные пиксели, лежащие во внутренней области.

Основная идея алгоритма с затравкой заключается в следующем:

Зная, что затравочный пиксель содержится в области, которую надо закрасить мы проверяем его соседей: каждый соседний пиксель, не являющийся граничным или уже закрашеным мы запоминаем как затравочный (то есть принадлежавший внутренней области и подлежащий закраске). Перебрав все пиксели таким образом мы сможем найти и закрасить все пиксели, принадлежащие внутренней области.

Однако очевидным решением является рекурсивный алгоритм, но так как использование рекурсии не целесообразно . Для преобразования рекурсивных алгоритмов используется стек . Вместо вызова в какой-то момент функции с другим объектом (как это происходит в рекурсии), мы запоминаем этот объект в стеке, чтобы обработать его на одной из следующих итераций.

Самый простой алгоритм заполнения с затравкой перебирает все пиксели “без разбора”: каждый сосед затравочного пикселя попадает в стек затравочный пикселей; как только рассмотрены все соседи текущего затравочного пикселя, из стека берется следующий затравочный пиксель (однако если он уже был обработан, то происходит просто переход к следующему в стеке пикселю) и так до тех пор, пока стек не будет пустым. Однако очевидно, что пиксели могут попадать в стек более 1 раза. В этом и есть главная проблема данного алгоритма: мы не знаем, как много памяти потребуется для хранения.

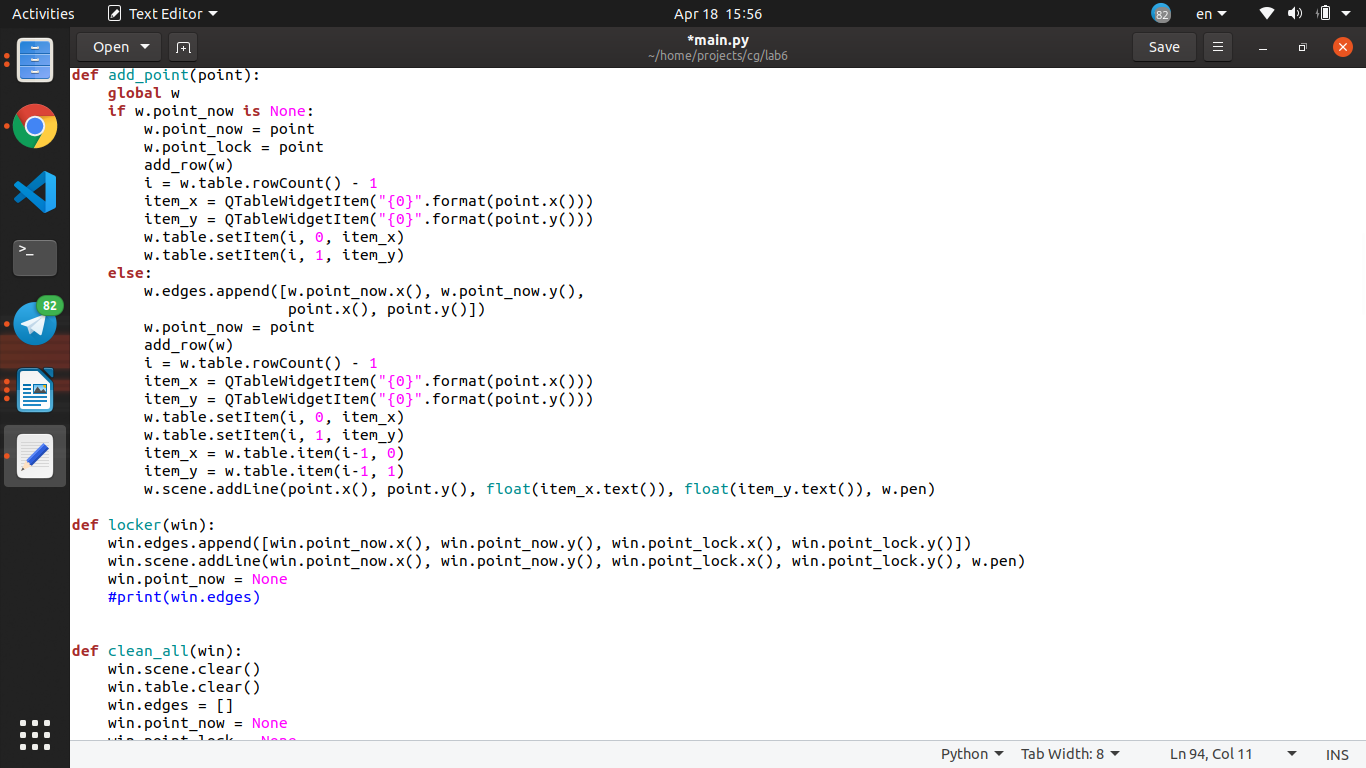
Проблему алгоритма выше решает *построчный алгоритм заполнения с затравкой.* В алгоритме выше стек содержит много ненужной и дублирующей информации. Поэтому в построчном алгоритме хранится только один затравочный пиксель для каждого непрерывного интервала на сканирующей строке. Теперь при рассмотрении пикселя делаются следующие действия:

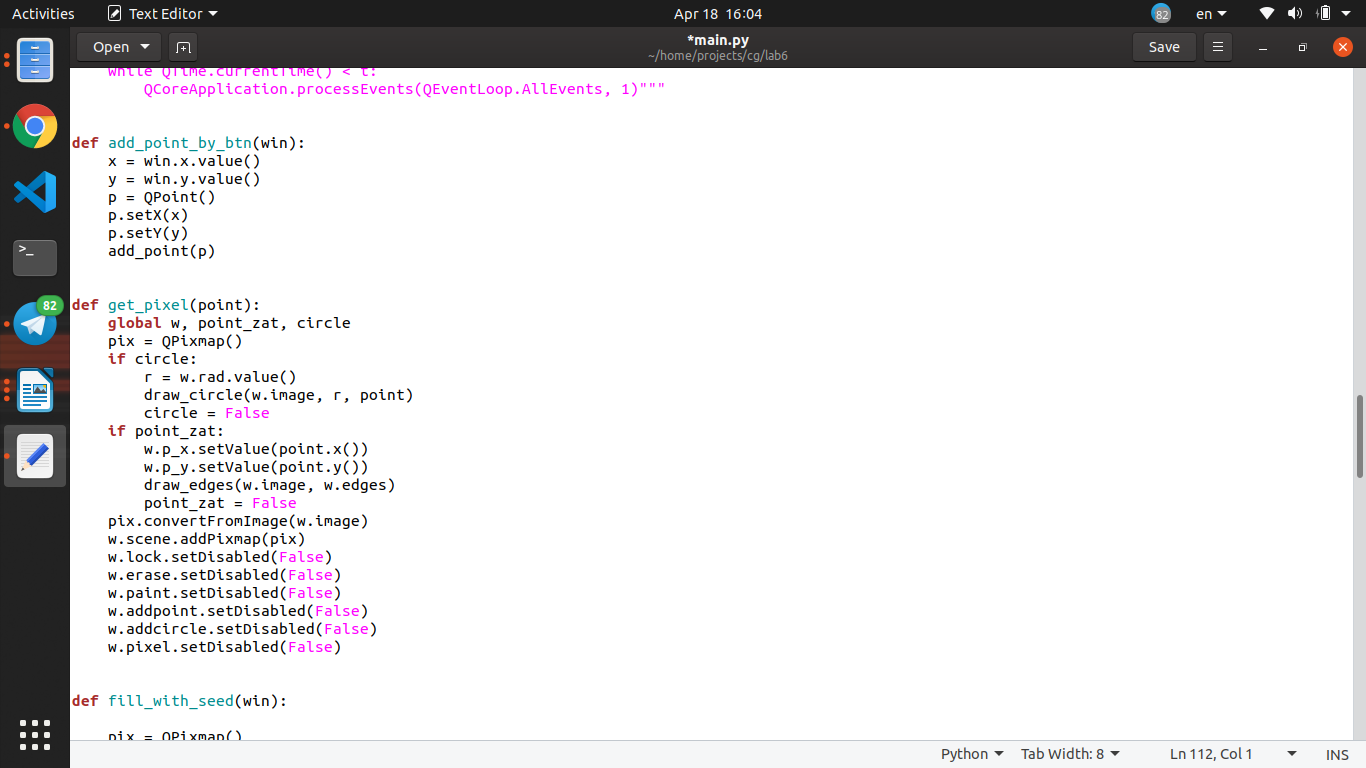
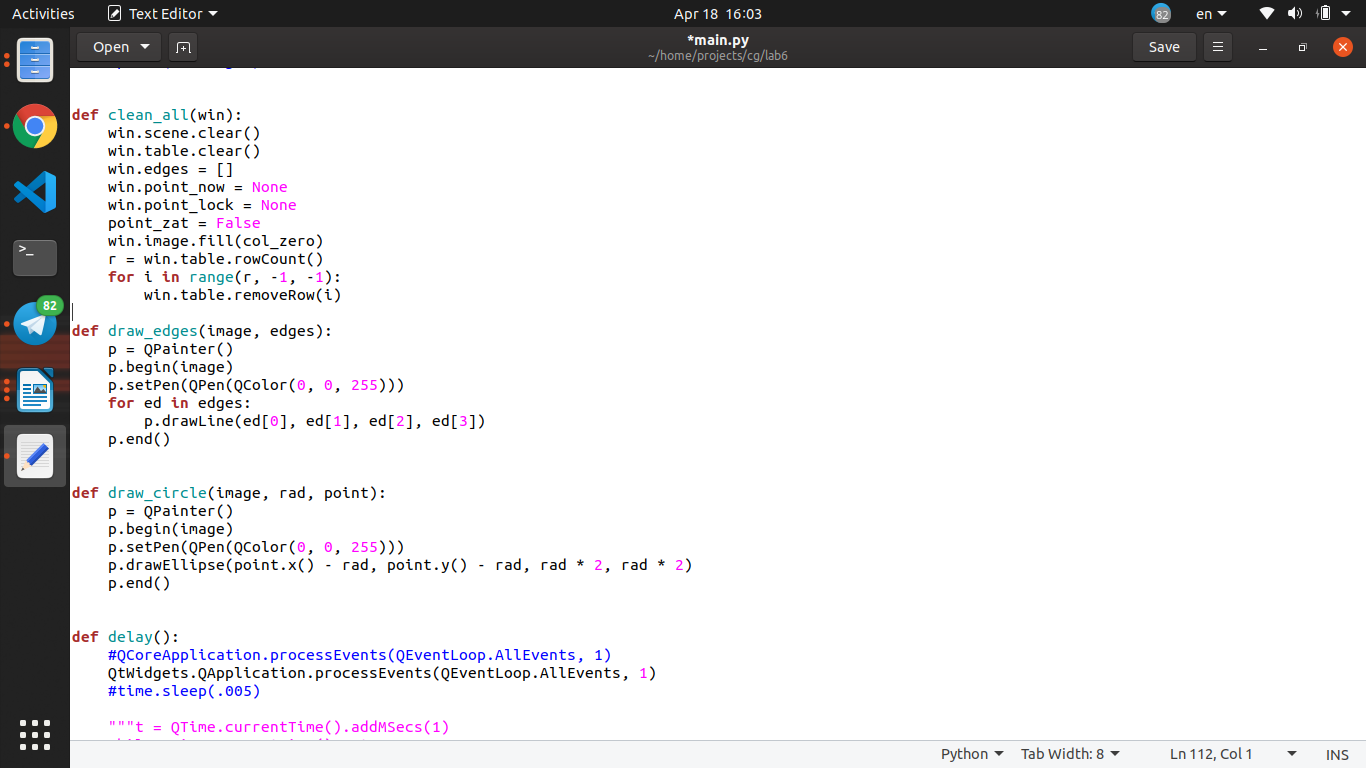
* От текущего затравочного пикселя помечаются все расположенные справа пиксели Доходим до граничного пикселя.

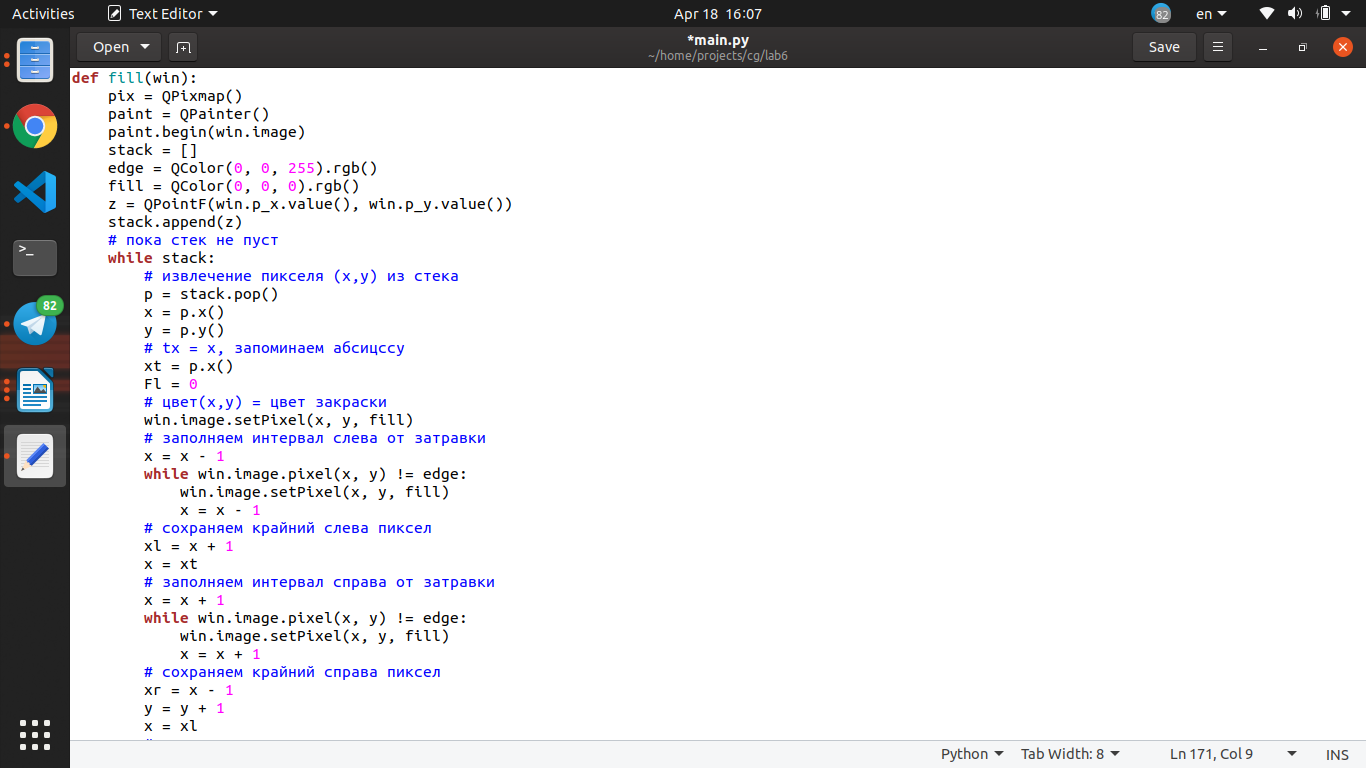
Запоминаем крайний правыйпиксель.

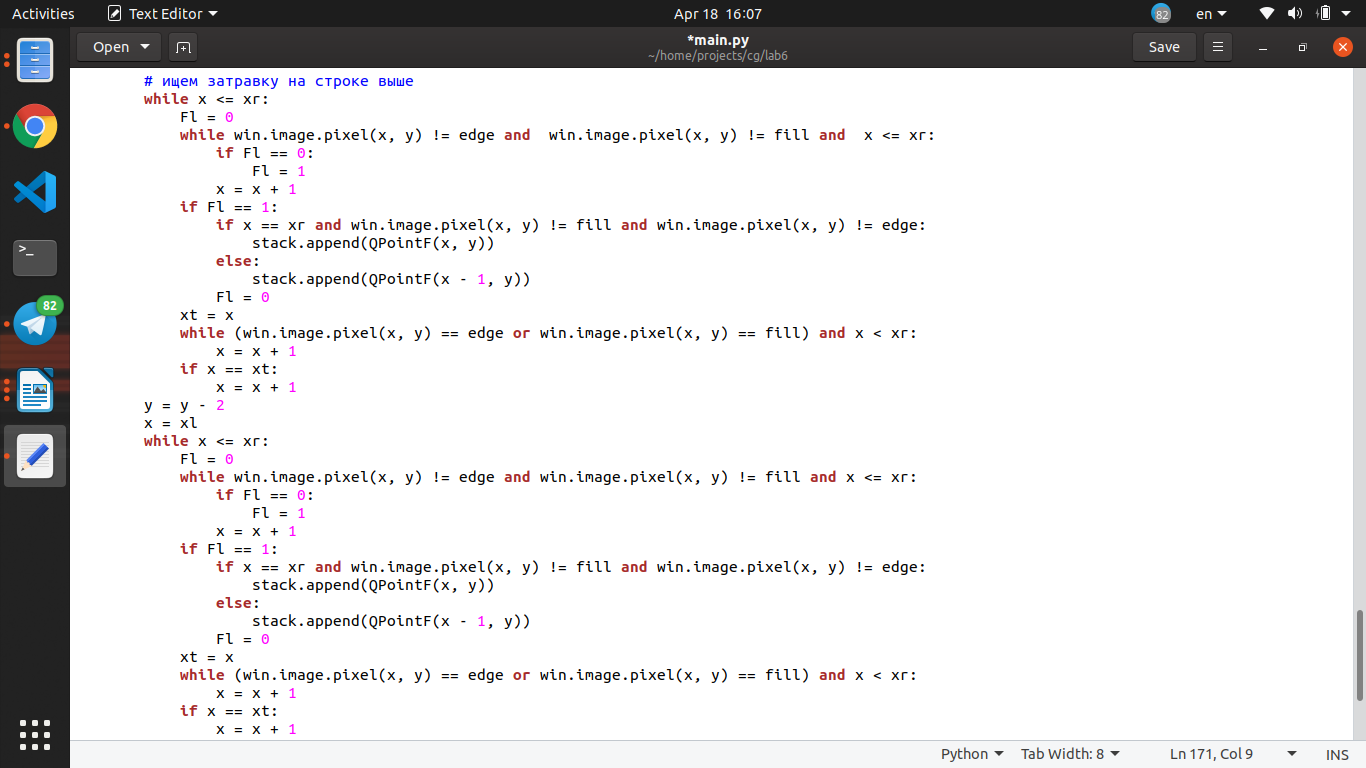
* Аналогично поступаем двигаясь в левом направлении. На данном шаге также запоминаем крайний левый пиксель.
* Рассматриваем верхнюю и нижнюю строки в интервале [Хлев: Хправ]. Во всех неперывных интервалах не закрашенных и не граничных пикселей, отмечаем самый правый пиксель.

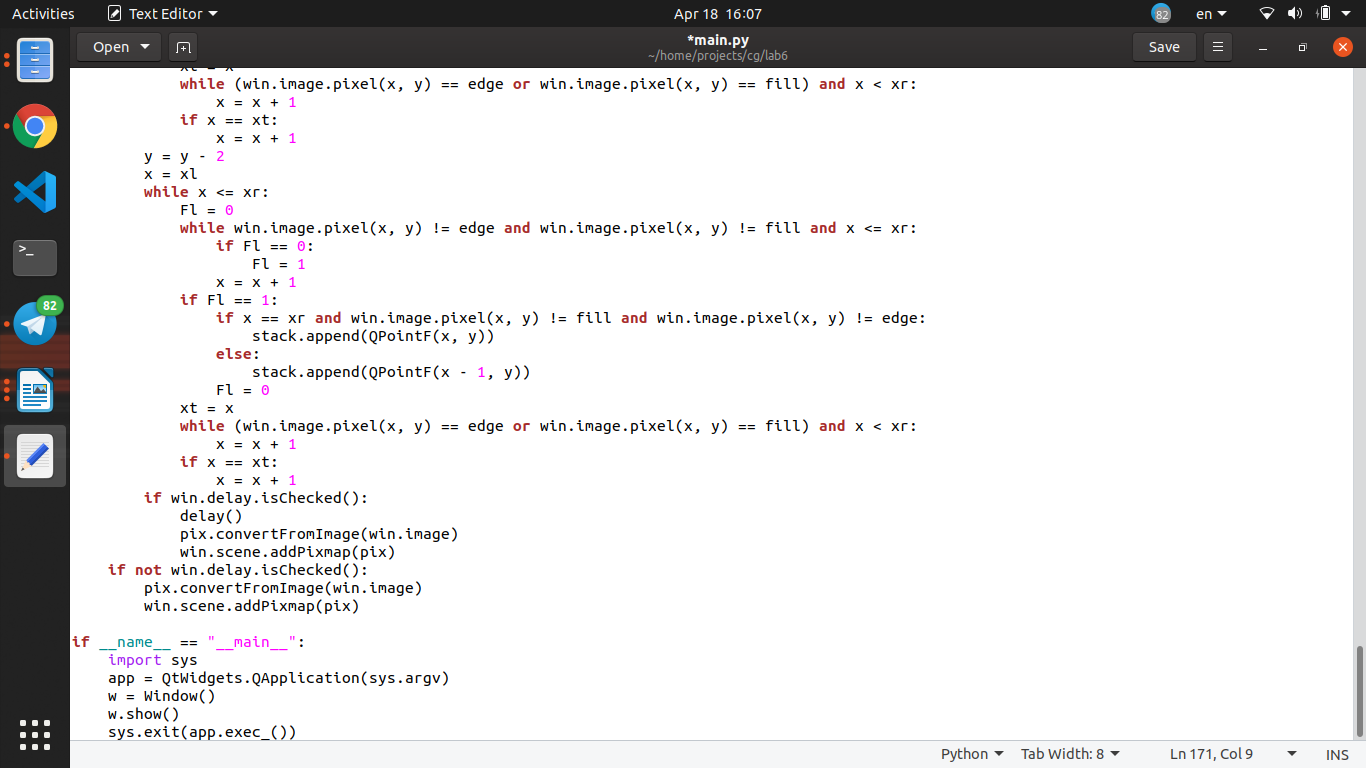
**Код программы**



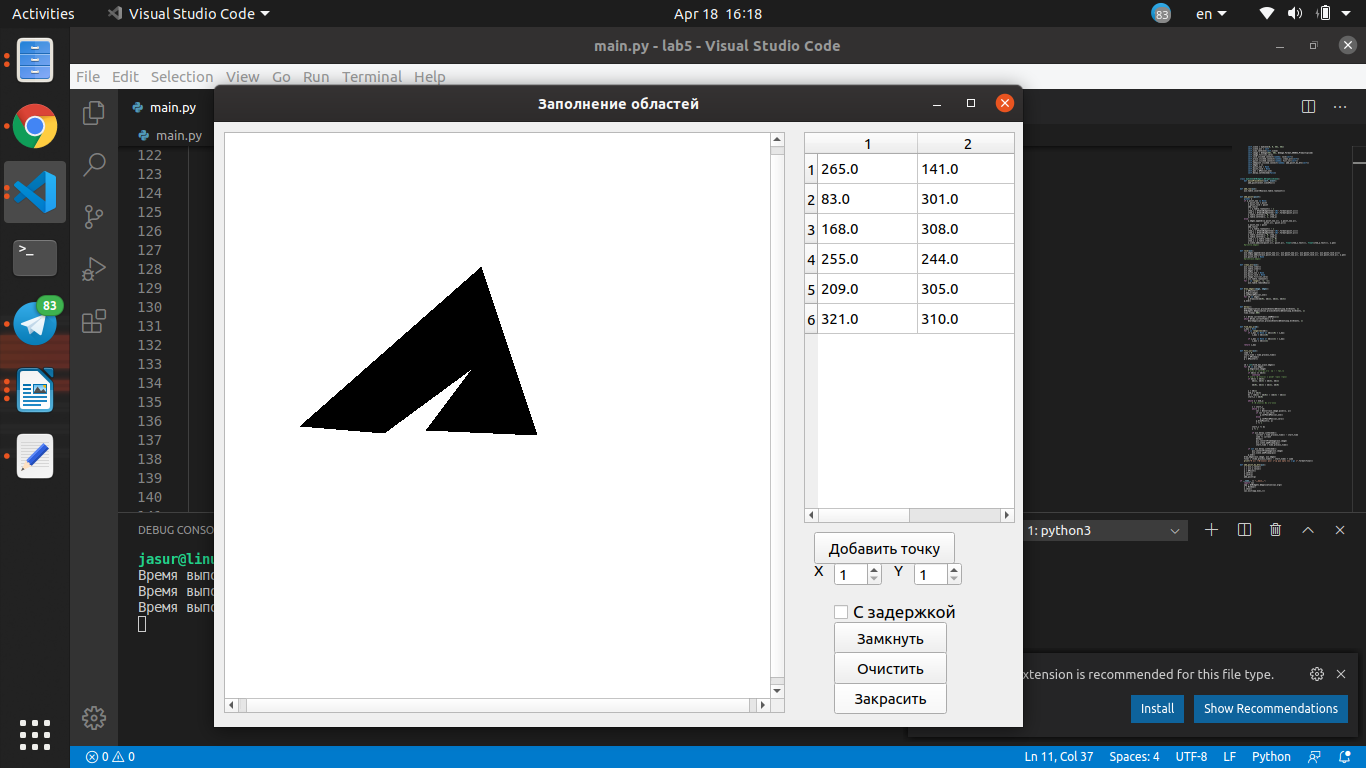
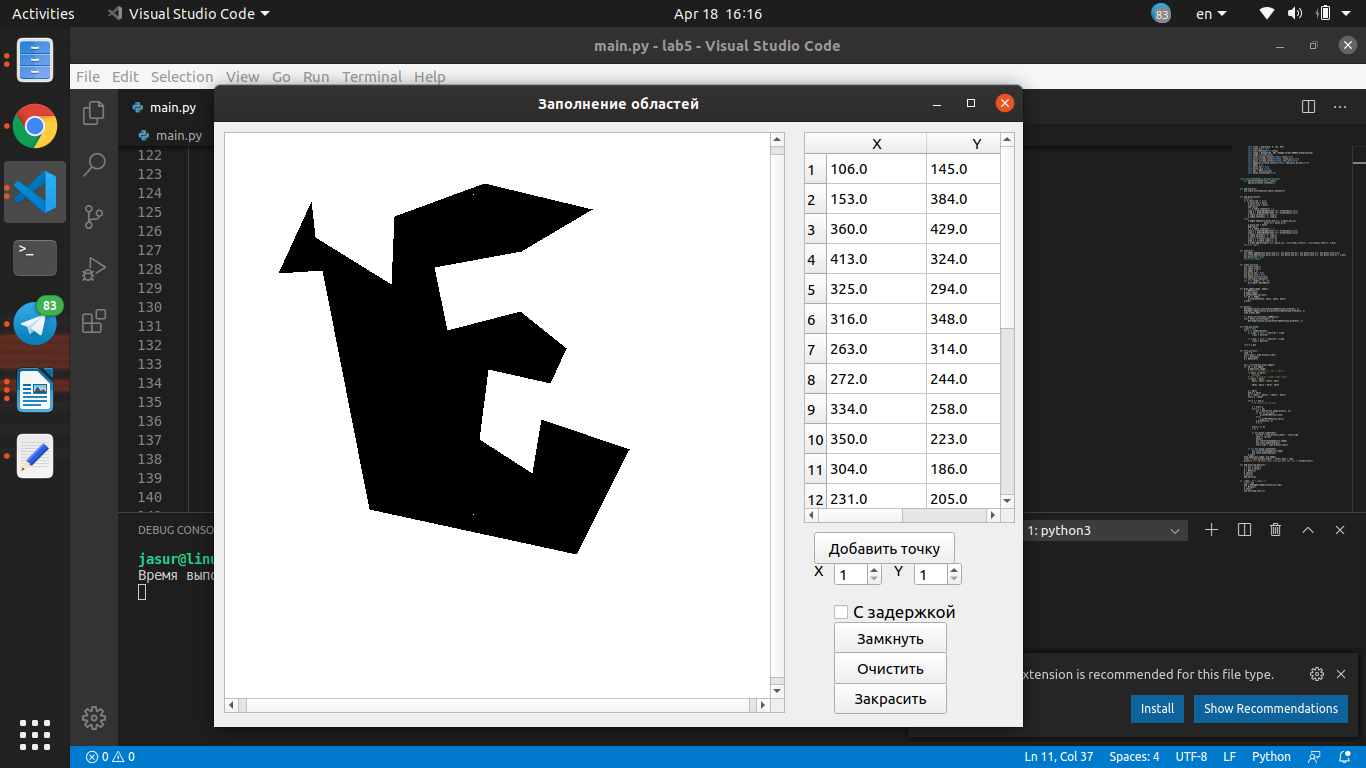


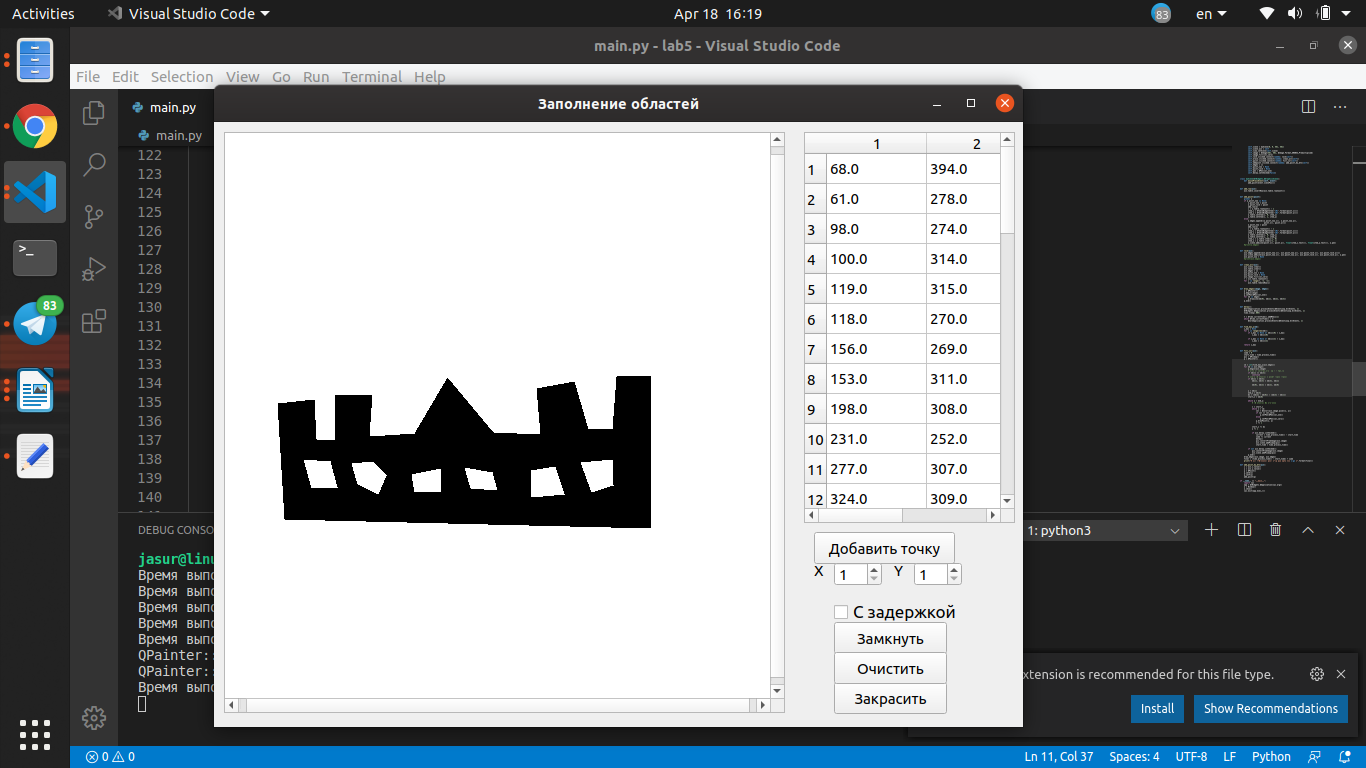
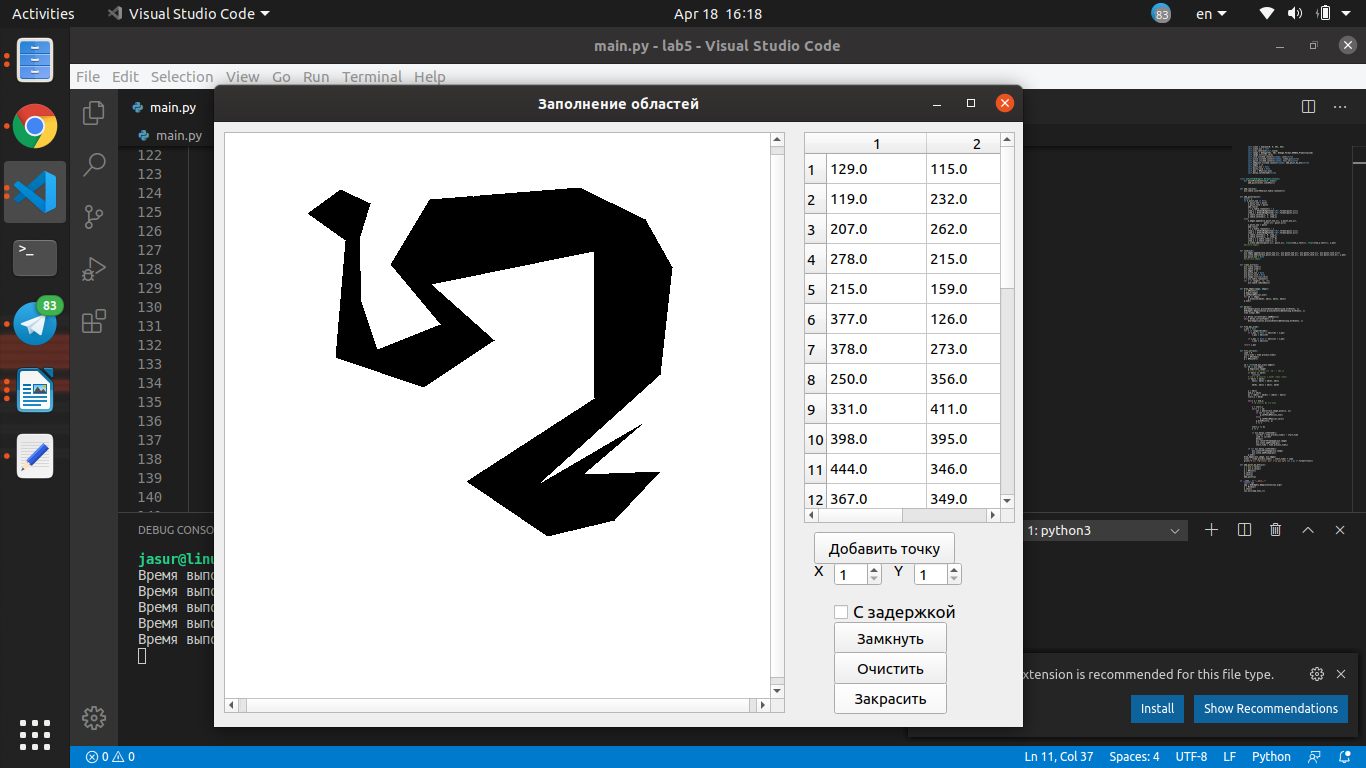


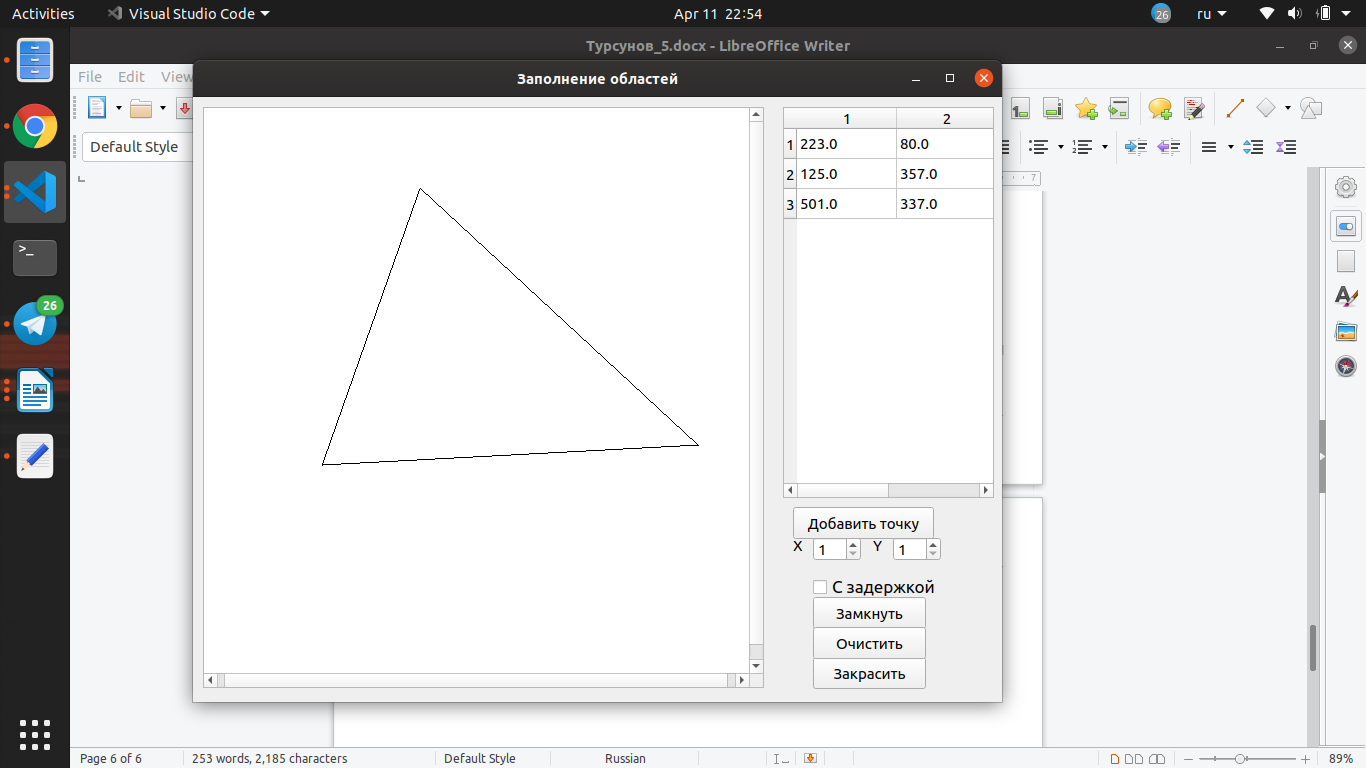




**Пример работы**

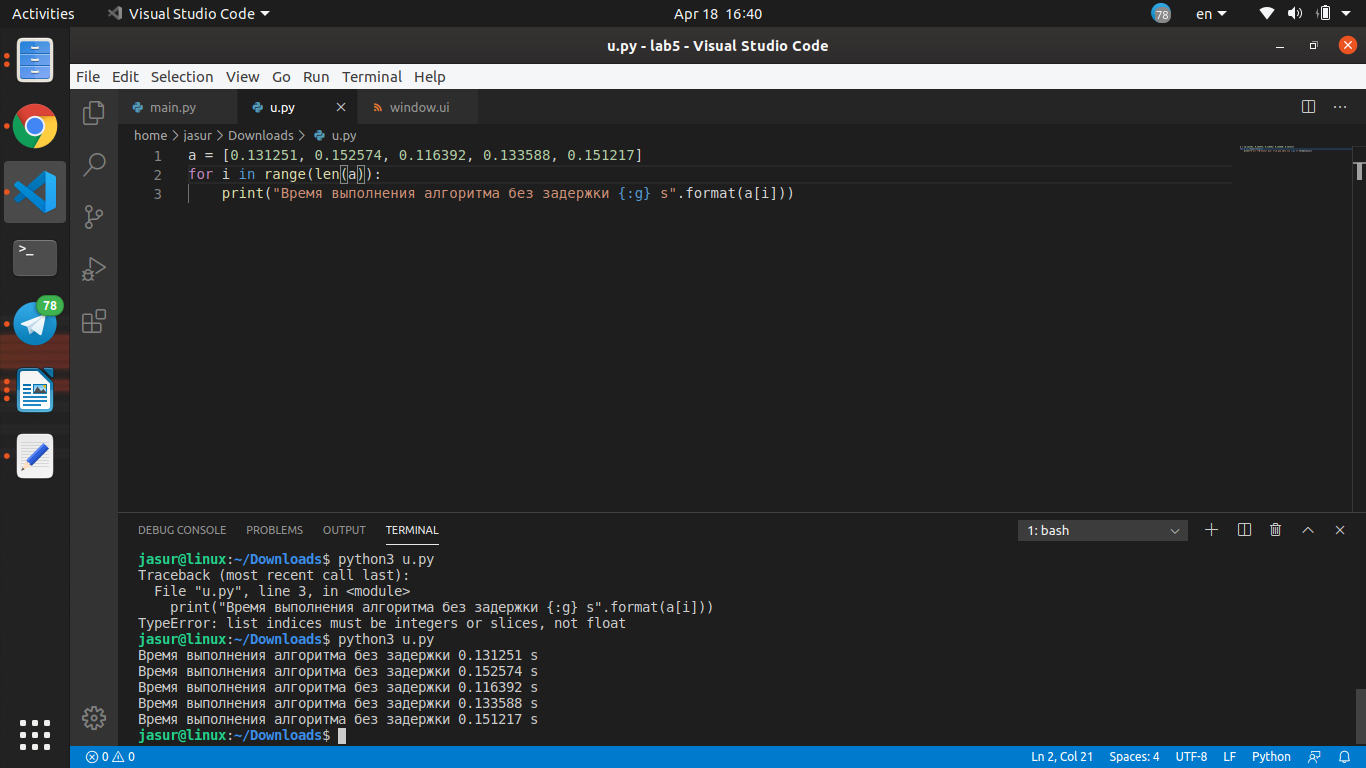




**Пользовательское меню**

**Замер времени**

Замер времени выполнения алгоритма (без задержки)

\*Время закраски многоугольников, рисунок которых указан выше(в секундах)