**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Московский технический университет связи и информатики**

Кафедра «Математическая кибернетика и информационные технологии»

**Курсовая работа**

по дисциплине «Введение в информационные технологии»

на тему «Подсчёт площади и определение выпуклости произвольного многоугольника»

Выполнил: студент группы БСУ1901

Панов Д. А.

Проверил: старший преподаватель кафедры МКиИТ

Мосева М.С.

Москва 2021

**Оглавление**

1. [Цель работы](file:///C:\Users\Даня\Desktop\Labs\Панов%20БСУ1901%20Курсовая%20Java.docx#_Toc91019364)

[2. Выполнение работы](file:///C:\Users\Даня\Desktop\Labs\Панов%20БСУ1901%20Курсовая%20Java.docx#_Toc91019370)

3. [Заключение](file:///C:\Users\Даня\Desktop\Labs\Панов%20БСУ1901%20Курсовая%20Java.docx#_Toc91019372)

**Цель работы**

Реализация данного алгоритма возможна путём создания двух функций. Первая для определения выпуклости, а вторая для нахождения площади.

Рассмотрим определение выпуклости. Как известно из школьного курса геометрии: Многоугольник называется выпуклым, если при проведении прямой через любую из его сторон весь многоугольник лежит только по одну сторону от этой прямой. Невыпуклыми являются все остальные многоугольники.

Многоугольник будет выпуклым если при его обходе в каждой тройке последовательных вершин происходит поворот всегда в одну и ту же сторону. Реализуя код тут будет необходимо, считав все координаты, пройтись по парам соседних сторон и определить знаки их векторного произведения. Если после просмотра всех пар соседних сторон встретились как положительные, так и отрицательные векторные произведения, то многоугольник не является выпуклым, иначе многоугольник считается выпуклым.

При нахождении площади многоугольника.

Пусть (х1, y1), (x2, у2), …, (хN,уN) —координаты вершин заданного многоугольника в порядке обхода по или против часовой стрелки. Тогда его ориентированная площадь S будет равна:

[Площадь многоугольника-5](https://gospodaretsva.com/wp-content/uploads/2012/03/34-5.png)

Рисунок 1 – Рабочая формула нахождения площади произвольно многоугольника.

Таким образом, основываясь на двух функциях, написанных по математической теории выше будет реализован алгоритм нахождения площади и определение выпуклости произвольного многоугольника.

В полной реализации алгоритма необходимо:

1. Создать функцию по подсчёту площади произвольного многоугольника

2. Написать функцию определения выпуклости многоугольника

3. Разработать интерфейс приложения.

Интерфейс должен быть удобен обычному пользователю. Кнопки должны быть расположены интуитивно для пользователя, цвета должны между собой сочетаться и быть приятными для глаза.

4. Написать функцию получения данных о координатах из интерфейса.

5. Написать функцию прорисовки многоугольника. Используя полученные ранее координаты точек.

6. Исключить ошибочный ввод. Прописать ошибки, при вводе не целого числа, ошибку при вводе менее 3 координат.

7. Позаботится об очистки данных и позволении повторного ввода для пользователя.

8. Создать надстройку графика. В моем случае это реализацию толщины линии.

9. Воплотить возможности сохранения данных, полученных при подсчёте алгоритмами, сохранение графика.

Таким образом, основная задача на исследование – это разработать удобное приложение для пользователя, которое могло бы просчитать площадь и определить выпуклость произвольного многоугольника.

# Описание разработанной программы

Ссылка на репозиторий в github:  
https://github.com/WouterLab/pyTerm\_geo

# Описание используемых библиотек

В ходе данной работы, мною были использованы две библиотеки, это: Tkinter и Python Imaging Library.

Tkinter – библиотека, позволяющая создавать удобный интерфейс для пользовательского приложения, она обладает множеством полезных и удобных функций. Например, создание кнопок и текстовых полей, есть возможность настроит фон приложения, создание полотна для рисования.

Отдельно из библиотеки Tkinter, так же был подключен раздел: messagebox, который позволяет программировать знакомые каждому всплывающие окна.

Python Imaging Library или сокращённо PIL – библиотека предназначенная для работы с растровой графикой. С помощью неё можно легко сохранить графические данные в формате png.

# Описание созданных функций

Основными функциями внутри моего приложения были подсчёт площади многоугольника (Листинг 1) и определение выпуклости многоугольника (Листинг 2).

Листинг 1 – Листинг функции «S\_function»

|  |
| --- |
| def S\_function():#подсчёт площади  file = open("file.txt", "a")  global x, y  cornersnumb= len(x)  S1 = 0  for i in range (cornersnumb):  Sum=x[i-1]\*y[i]-x[i]\*y[i-1]  S1+=Sum  S = float()  S = abs(0.5 \* S1)  file.write('Площадь полученного многоугольника: ')  S = str(S)  file.write(S)  text = Label(win, text = str(S), width=25, background="#E6E6FA").grid(row=20, column=4, stick ="w") |

Данная функция подсчитывает площадь многоугольника, используя координаты его вершин по формуле, представленной на рисунке 1.

Изначально x и y – это два массива данных, которые содержат внутри себя данные об x координатах точки и y соответственно. С помощью цикла «for» мы проходим через каждую пару координат для подсчёта площади многоугольника.

Листинг 2 – Листинг функции «vyp\_function»

|  |
| --- |
| def vyp\_function():#определение выпуклости    file = open("file.txt", "a")  file.write(' \n')    global x, y  cornersnumb= len(x)  PosAngle = False  NegAngle = False  for i in range(cornersnumb):  x1 = x[i]  y1 = y[i]  x2 = x[(i+1) % cornersnumb]  y2 = y[(i+1) % cornersnumb]  x3 = x[(i+2) % cornersnumb]  y3 = y[(i+1) % cornersnumb]  d = (x2 - x1) \* (y3 - y2) - (y2 - y1) \* (x3 - x2)  if d > 0:  PosAngle = True  elif d < 0:  NegAngle = True  if PosAngle and NegAngle:  res = 'Многоугольник не выпуклый'  file.write(res)  text = Label(win, text = str(res), width=25, background="#E6E6FA").grid(row=22, column=4)  else:  res = 'Многоугольник выпуклый'  file.write(res)  text = Label(win, text = str(res), width=25, background="#E6E6FA").grid(row=22, column=4) |

Данная функция позволяет определить является ли многоугольник выпуклым. Функция получает данные о глобальных переменных списках x и y и работает непосредственно с ними. Функция работает следующим образом: необходимо пройти по всем парам соседних сторон (не забывая про зацикливание) посмотреть на знаки векторного произведения.

Фрагмент функции «vyp\_function»:

|  |
| --- |
| d = (x2 - x1) \* (y3 - y2) - (y2 - y1) \* (x3 - x2) |

Следует отмечать, какие по знаку векторные произведения встречались: положительные (переменная PosAngle) и отрицательные (переменная NegAngle), нулевые произведения игнорируются. Если после просмотра всех пар соседних сторон встретились как положительные, так и отрицательные векторные произведения, то многоугольник не является выпуклым и функция выводит «Многоугольник не выпуклый», иначе функция выводит на экран пользовательского приложения «Многоугольник выпуклый».

В программе так же присутствует функция, которая рисует многоугольник. Код данной функции представлен на Листинге 3.

Листинг 3 – Листинг функции «draw\_function»

|  |
| --- |
| def draw\_function():#построение линий на графике  global x, y  cornersnumb = len(x)  if (cornersnumb < 3):  message=messagebox.showerror(("Ошибка"), ("Многоугольник не может содержать менее 3 углов"))  else:  try:  tolsh=int(tol.get())  except ValueError:  message=messagebox.showerror(("Ошибка"), ("Введите целое число"))  for i in range (cornersnumb):  canvas.create\_line((x[i-1]\*50), (y[i-1]\*50), (x[i]\*50), (y[i]\*50), width = tolsh, fill="red") |

Внутри этой функции присутствует и проверка на количество вводимых координат. Если пользователь введёт менее 3 координат (x,y), то программа не позволит построить такую фигуру и всплывёт окно с ошибкой.

Фрагмент функции «draw\_function»:

|  |
| --- |
| if (cornersnumb < 3):  message=messagebox.showinfo(("Ошибка"), ("Многоугольник не может содержать менее 3 углов")) |

Если же пользователь введёт 3 и более координат, то программа построит на холсте интерфейса фигуру.

Надстройка управления графиком, на котором строится фигура, также приведено здесь. Пользователь может ввести свою толщину для линии на графике.

В программе так же предусмотрена функция, которая позволяет отчистить всё. Она представлена на Листинге 4.

Листинг 4 – Листинг функции «del array»

|  |
| --- |
| def del\_array():#отчистка массивов, содержащих внутри себя данные, введённых точек; отчистка холста  file = open("file.txt", "w")  file.write('Введённые координаты вершин многоугольника: \n')  x.clear()  y.clear()  canvas.delete("all")  fieldx.delete(1.0, "end")    for i in range(10):  canvas.create\_line(i\*50, 0, i\*50, 500, fill= "grey") #вертикальные линии  canvas.create\_text(i\*50, 10, text= i )  canvas.create\_line(0, i\*50, 500, i\*50, fill= "grey") #горизонтальные линии  canvas.create\_text(10, i\*50, text=i)  canvas.create\_line(2, 2, 500, 2, fill= "red", arrow=LAST)  canvas.create\_line(2, 2, 2, 500, fill= "red", arrow=LAST)  canvas.create\_line(2, 500, 500, 500, fill= "red")  canvas.create\_line(500, 500, 500, 2, fill= "red")  canvas.grid(row =16, rowspan = 6, column=0, columnspan=6, sticky="w")  canvas.create\_text(10,10, text="0,0") |

Эта функция отчищает данные, содержащиеся в массиве и отчищает холст, на котором мог быть уже построен график.

Функция «on\_closing» спрашивает у пользователя можно ли завершить работу с программой, при закрытии приложения. Она представлена на Листинге 5.

Листинг 5 – Листинг функции «on\_closing»

|  |
| --- |
| def on\_closing():  if messagebox.askokcancel("Выход из приложения", "Хотите выйти из приложения?"):  win.destroy() |

Эта функция позволяет избежать случайного закрытия окна пользователем во время работы.

Последняя функция, которая ещё не была описана «get\_all». Функция представлена на Листинге 6.

Листинг 6 – Листинг функции «get\_all»

|  |
| --- |
| def get\_all():  try:  valuex = int(xname.get())  except ValueError:  message=messagebox.showinfo(("Ошибка"), ("Введите число"))  x.append(valuex)  print('x:',x)  try:  valuey = int(yname.get())  except ValueError:  message=messagebox.showinfo(("Ошибка"), ("Введите число"))  y.append(valuey)  print('y:',y) |

Данная функция получает данные вводимые пользователем из приложения и проверяет являются ли они числами, иначе всплывает ошибка, которая предупреждает пользователя о необходимости вводить числа.

Функция для сохранения полученных значений представлена на Листинге 7.

|  |
| --- |
| def save():  x=win.winfo\_rootx()+canvas.winfo\_x()+8  y=win.winfo\_rooty()+canvas.winfo\_y()+80  x1=x+canvas.winfo\_width()+120  y1=y+canvas.winfo\_height()+110  ImageGrab.grab().crop((x,y,x1,y1)).save("D:\Polly\proramming\TP/file.png") |

Данная функция позволяет пользователю сохранить график построенной фигуры в формате «png». Что очень удобно при громоздких расчётах.

В таблице 1 приведены данный о всех переменных, использованных в программе, их название, тип переменной, и назначение переменной.

Таблица 1 – Входные и выходные данные

|  |  |  |
| --- | --- | --- |
| Переменная | Тип переменной | Её значение |
| x | list | Хранит в себе значение координат «x» каждой точки |
| Продолжение Таблицы 1 | | |
| y | list | Хранит в себе значение координат «y» каждой точки |
| valuex | int | Переменная, которая проверяет является ли вводимая координата целым числом |
| valuey | int | Переменная, которая проверяет является ли вводимая координата целым числом |
| message | str | Сообщение об ошибке |
| cornersnumb | int | Количество координат в списке |
| S1 | Int | Отдельная просчитанная пара координат |
| Sum | Int | Сумма общая по формуле площади многоугольника |
| S | Float | Площадь многоугольника |
| text | str | Результат подсчета площади (выводится на экран) |
| PosAngle | bool | Положительные направления ребра многоугольника |
| NegAngle | bool | Отрицательные направления ребра многоугольника |
| res | str | Результат выпуклости (выводится на экран) |
| win | Tk | Окно приложения |
| Bg\_logo | label | Фоновое изображение приложения |
| Продолжение Таблицы 1 | | |
| canvas | canvas | Холст, на котором происходит прорисовка многоугольника |
| Zagolovokcanvas | label | Заголовок холста |
| Zagolovokx | label | Заголовок у координаты x |
| Zagolovoky | label | Заголовок у координаты y |
| xname | entry | Поле ввода координаты x |
| yname | entry | Поле ввода координаты y |
| file | textIOwrapper | Текстовый документ, в котором сохраняются полученные и введенные значения |
| tol | grid | Поле для ввода толщины линии |
| Zagolovoktol | label | Заголовок при вводе толщины линий графика |
| x | int | Данные о пикселе где располагается холст |
| y | int | Данные о пикселе где располагается холст |
| X1 | int | Данные о пикселе где располагается холст |
| Y1 | int | Данные о пикселе где располагается холст |

На рисунке 2 представлен Интерфейс приложения.

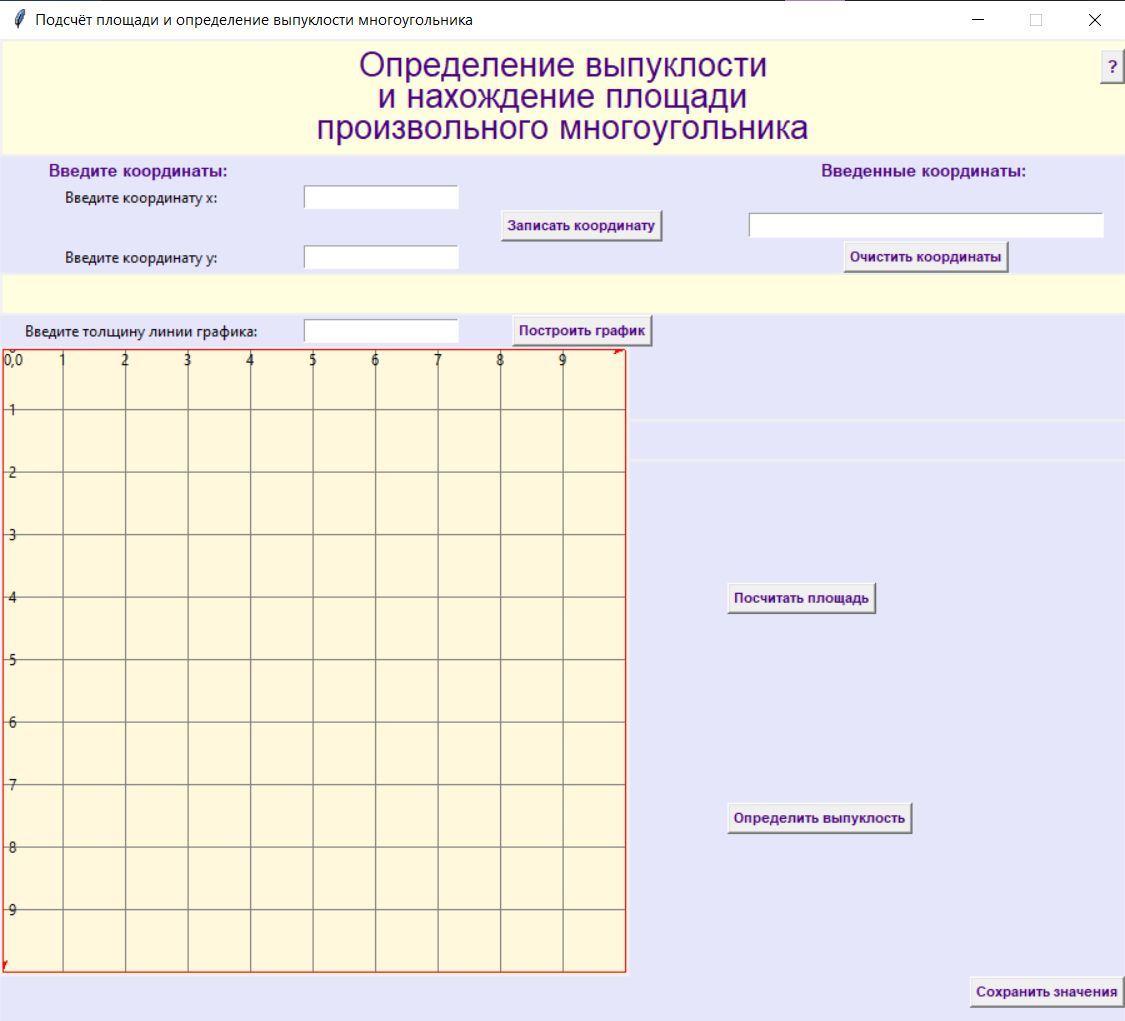


Рисунок 2 – Интерфейс приложения «Подсчёт площади и определение выпуклости производного многоугольника»

# Реализация приложения и Тестирование программы

Приложение было реализовано для подсчёта площади произвольного многоугольника и определения его выпуклости. Интерфейс приложения представлен на рисунке 2.

На главном экране пользователь видит несколько кнопок, холст для прорисовки фигуры и несколько полей для ввода данных.

Изначально необходимо по очереди ввести координаты углов многоугольника (например, в поле для ввода координаты x вводим первую цифру (значение координаты x и y обязательно должно быть числами), в поле координаты y вводим y) и нажимаем кнопку «Записать координату».

Если пользователь будет вводить координаты в поля ввода координат, которые не являются числами, тогда появится ошибка, показанная на рисунке 3.

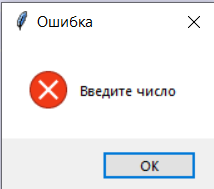


Рисунок 3 – Ошибка не верного ввода

Координаты отобразятся в поле «Введенные координаты: », потом в поля x и y вводим данные о следующей координаты. Нужно помнить, что координаты нужно вводить в порядке по часовой или против часовой стрелки, иначе фигура будет построена неправильно. Далее после ввода минимум трёх координат (иначе программа не позволит построить фигуру и выдаст ошибку, представленную на рисунке 4, ведь многоугольник может содержать минимум три координаты), пользователю необходимо ввести переменную в отведенное ей поле, которая будет иметь значение толщины линий и нажать кнопку «Построить график» и на холсте появится построенная по вводимым координатам ранее фигура.

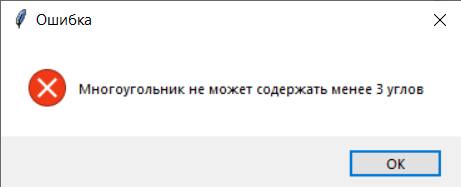


Рисунок 4 – Ошибка при вводе менее 3 углов

На данном этапе предлагается проверить пользователю правильность фигуры, если координаты были введены неправильно, то предлагается нажать на кнопку «Очистить координаты» (тогда списки координат и холст отчистятся) и повторить ввод заново.

Далее, когда пользователь убедился в правильности построенной фигуры, необходимо нажать поочередно 2 кнопки «Посчитать площадь многоугольника» и «Определить выпуклость». После этого на экране приложения появятся подсчитанная площадь многоугольника и является ли он выпуклым.

После получения всех построений стоит нажать на кнопку «Сохранить значения» и программа сохранит график в формате png на компьютер пользователя.

На рисунке 5 представлен пример работы программы.

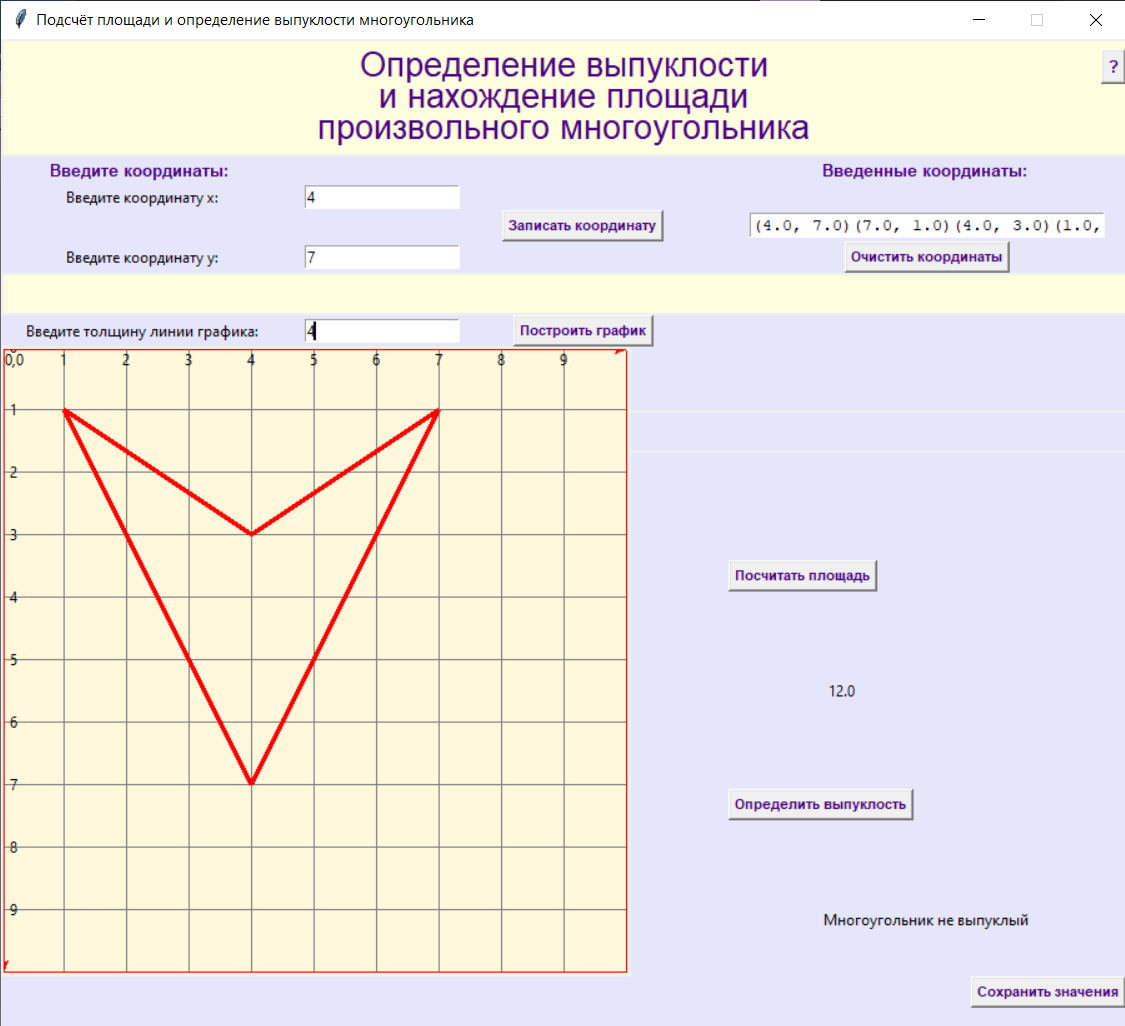


Рисунок 5 – Пример работы программы

Сохраненный в формате png график представлен на рисунке 6.

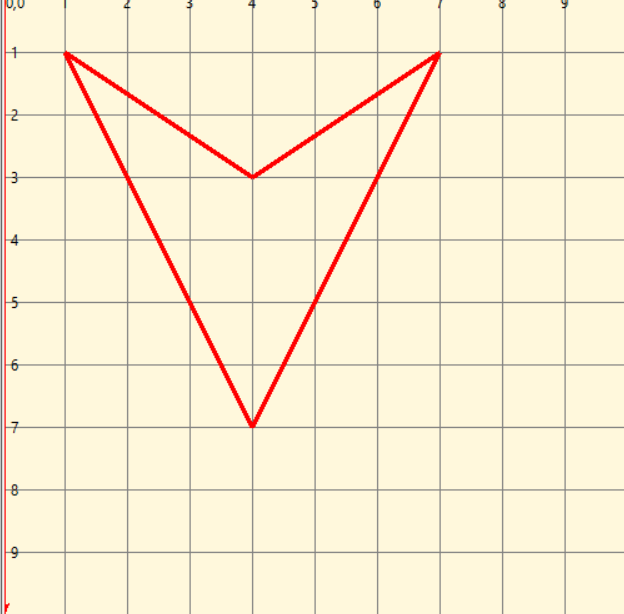


Рисунок 6 – Сохраненный в формате «png» график

На рисунке 7 представлены сохранённые данные в формате «txt».

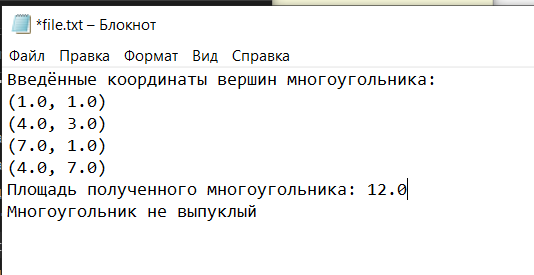


Рисунок 7 – Сохраненные данные в формате «txt»

По завершению работы, необходимо нажать на крестик, тогда появится следующее всплывающее окно, представленное на рисунке 8.

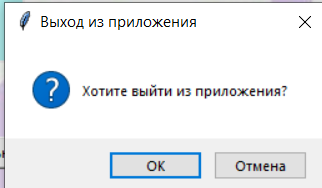


Рисунок 8 – Уведомление о закрытии приложения

Программа была протестирована успешно, все функции выполняются правильно. Ошибки возможны только при построении пользователем неправильных фигур. Избежать этого можно при правильном порядке ввода координат по часовой стрелке или против.

В таблице 2 представлены тесты программы. Тестовый случай, входные данные, ожидаемый результат и результат.

Таблица 2 – Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый случай | Значение вводимых данных | Ожидаемый результат | Результат программы |
| Рисунок 9 | X: 0,25; 1,25; 1,25; 0,25  Y: 0,25; 0,25; 1,25; 1,25 | 1.0  Многоугольник выпуклый | 1.0  Многоугольник выпуклый |
| Рисунок 10 | X: 1  Y: 1 | Ошибка:  Многоугольник не может содержать менее 3 углов | Ошибка:  Многоугольник не может содержать менее 3 углов |
| Рисунок 11 | X: k  Y:100 | Ошибка:  Введите целое число | Ошибка:  Введите целое число |
| Продолжение таблицы 2 | | | |
| Рисунок 12 | X: 0,25; 1,25; 0,75; 1,25; 0,25 Y: 0, 25; 0,25; 0,75; 1,25; 1,25 | 0,75  Многоугольник не выпуклый | 0,75  Многоугольник не выпуклый |
| Рисунок 13 | X: 1; 4; 4; 1  Y: 1; 1; 4; 4 | 9  Многоугольник выпуклый | 9  Многоугольник выпуклый |

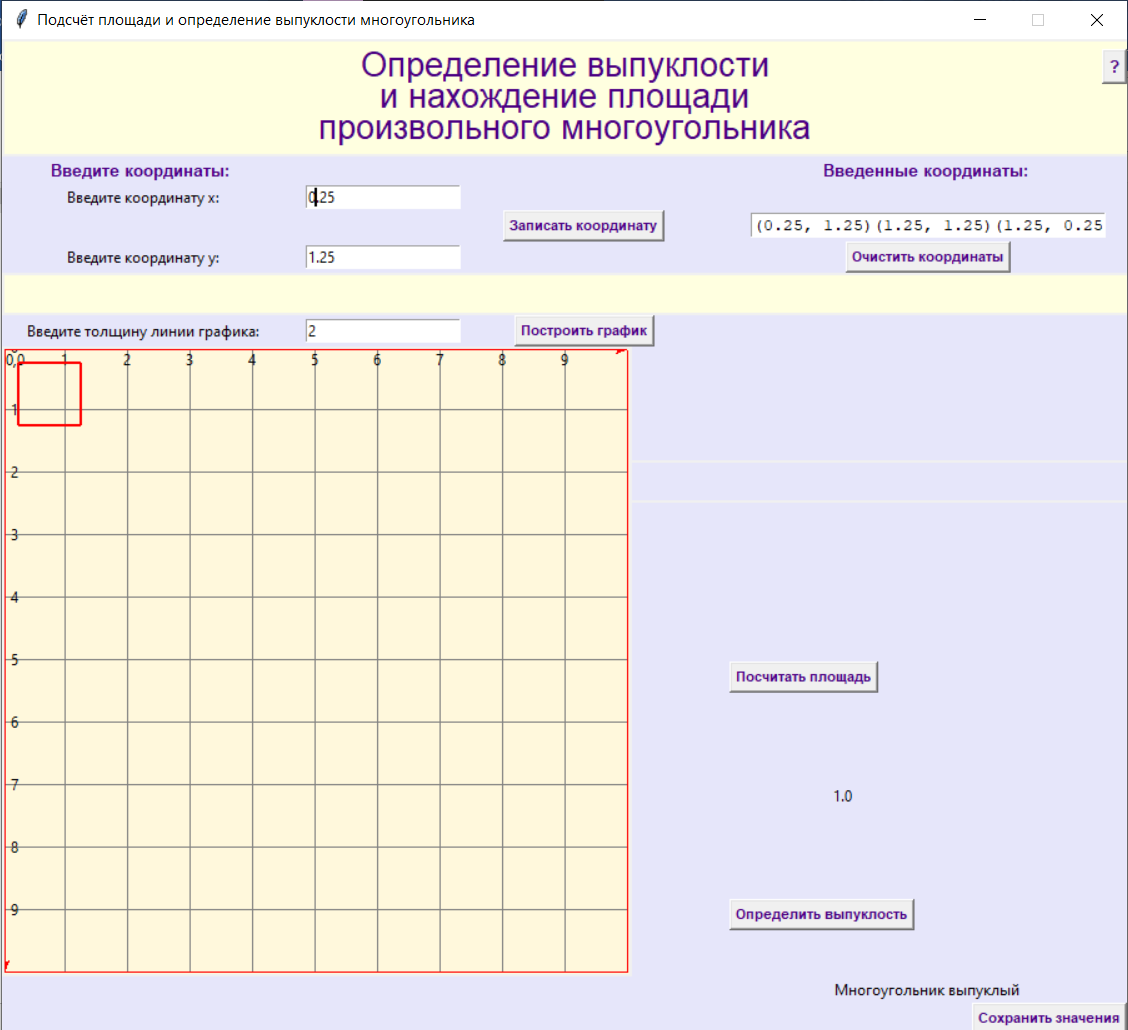


Рисунок 9 - Тестовый случай 1

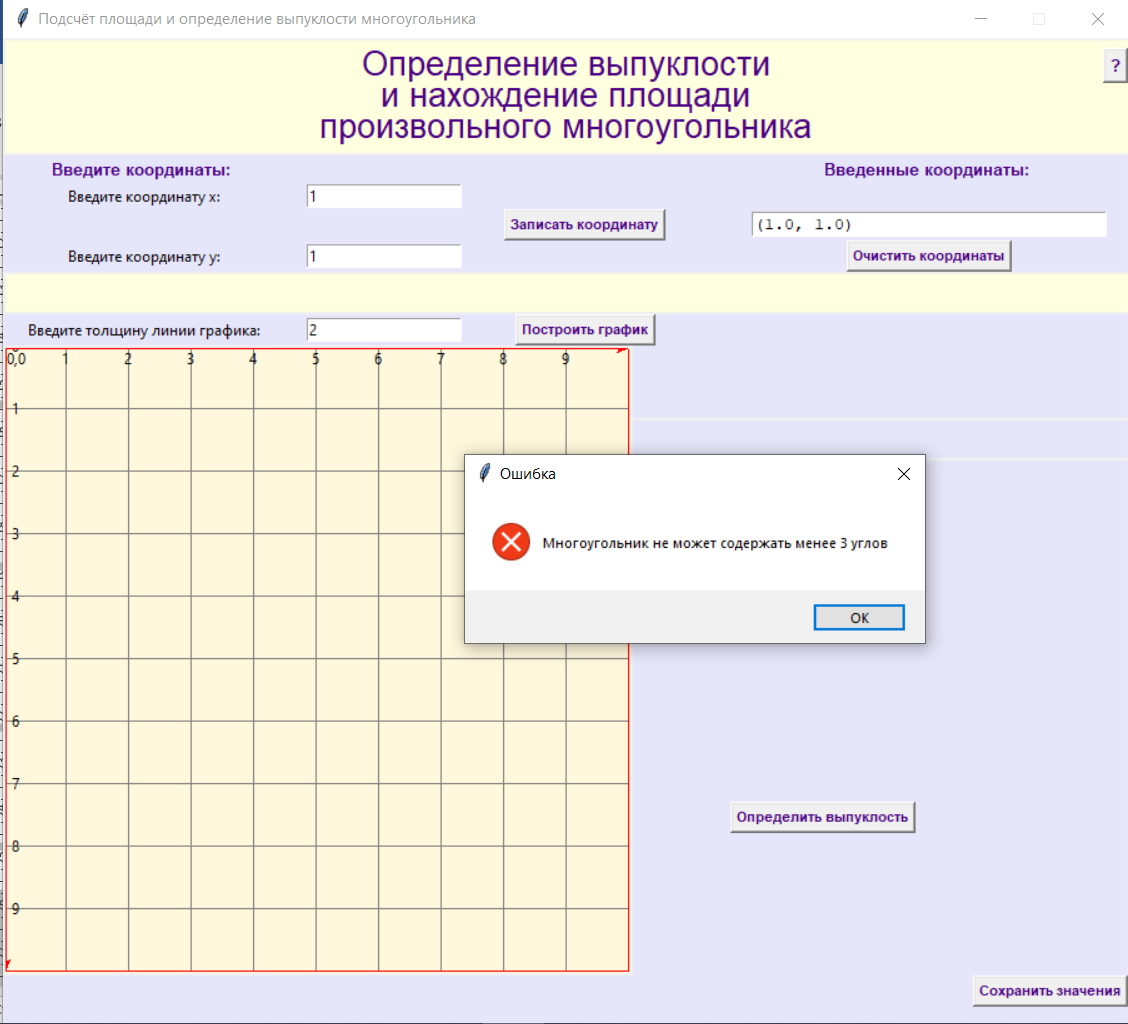


Рисунок 10 – Тестовый случай 2

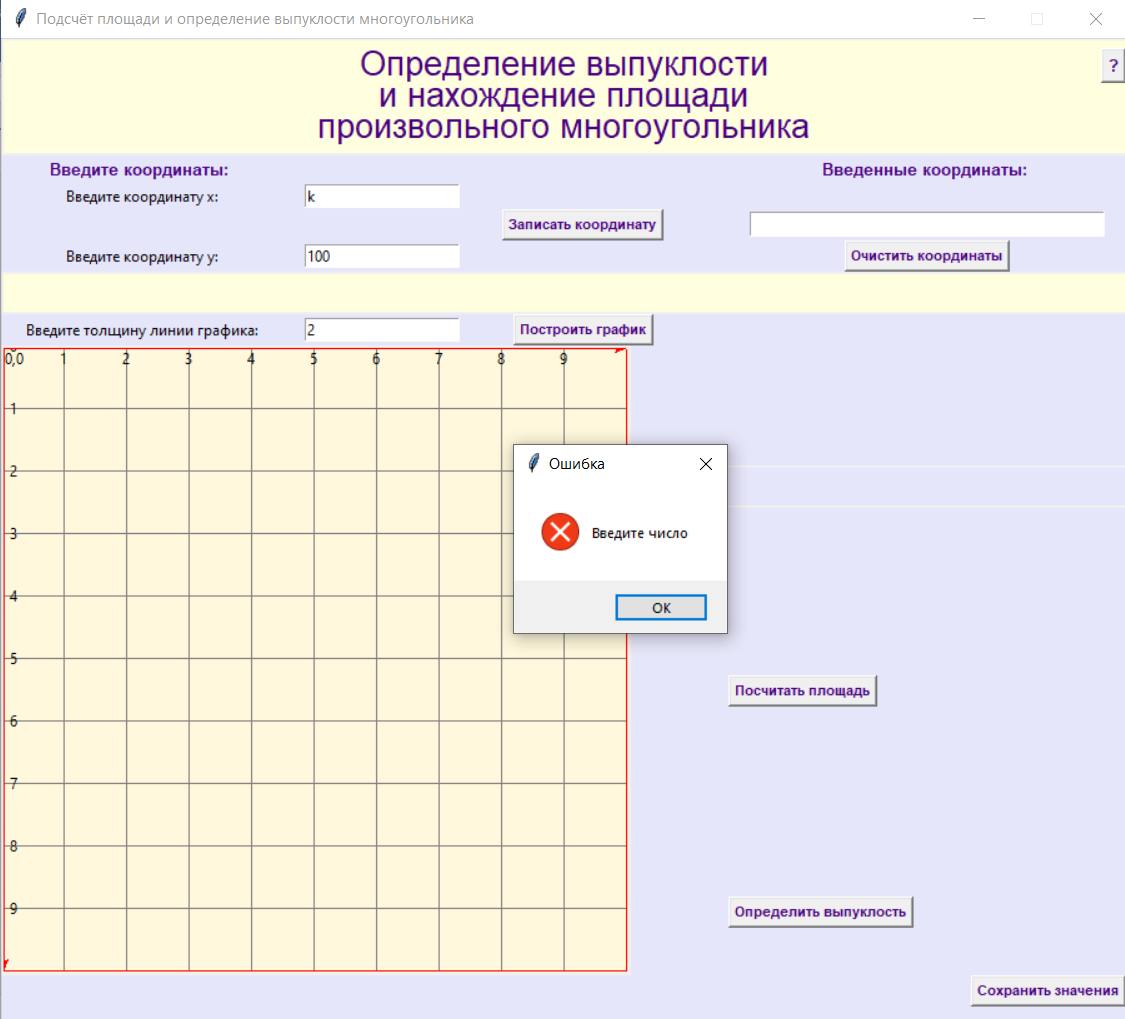


Рисунок 11 – Тестовый случай 3

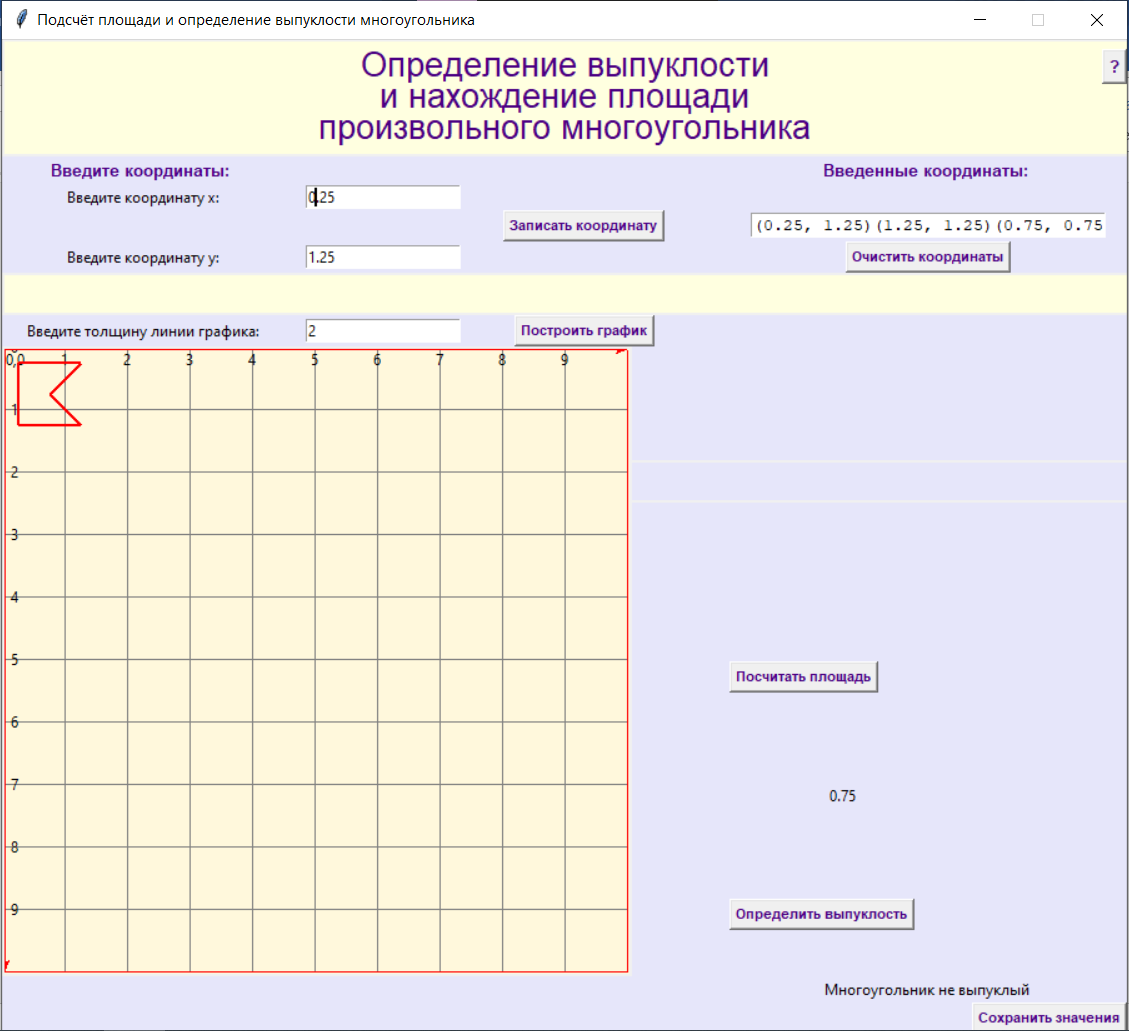


Рисунок 12 – Тестовый случай 4

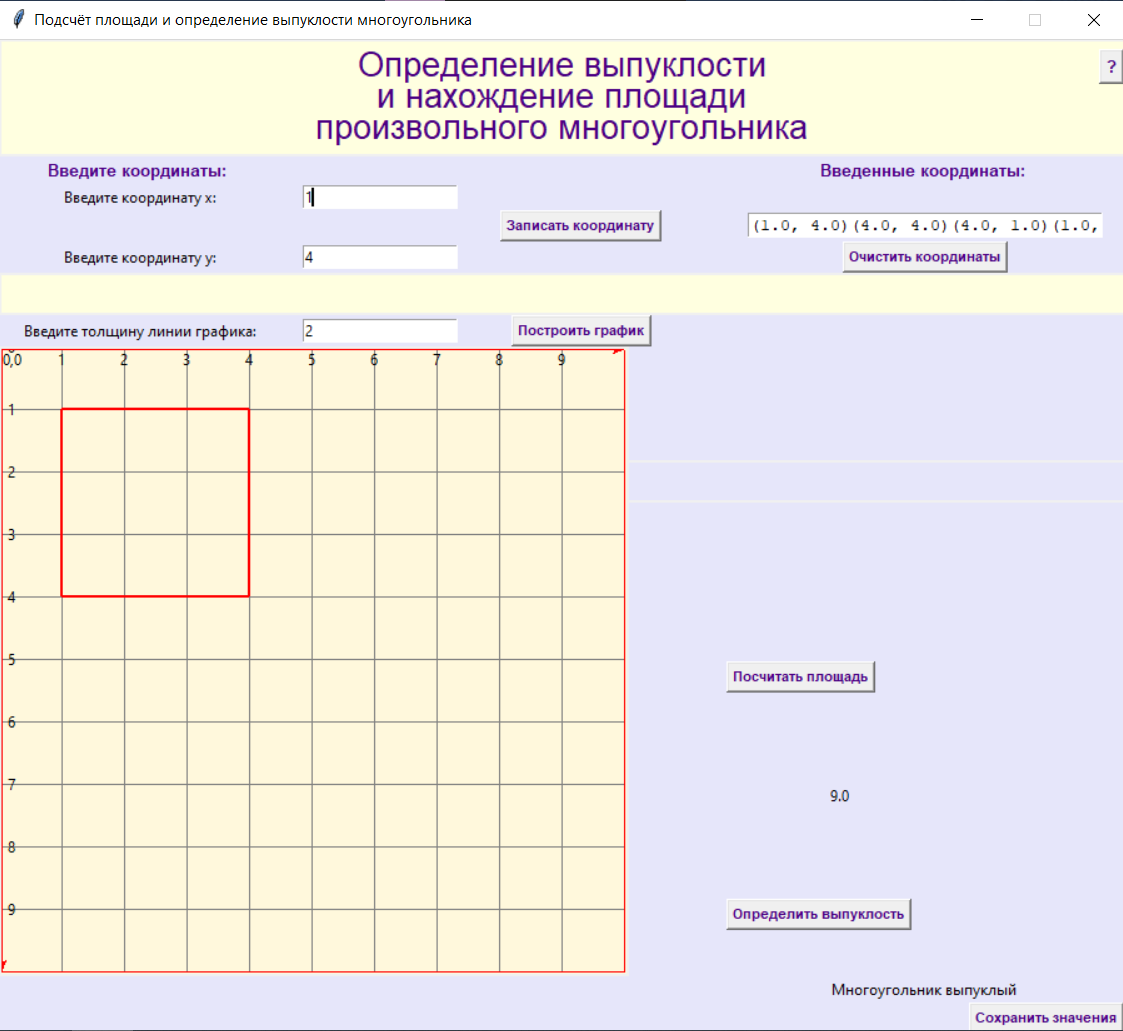


Рисунок 13 – Тестовый случай 5

# Заключение

Таким образом, в ходе выполнения данной курсовой работы, после анализа и изучения тем выпуклости и площади произвольного многоугольника, было разработано приложение на языке Python с использованием библиотек Tkinter b PIL. Приложение имеет удобный и интуитивный для пользователя интерфейс, оно оснащено уведомлениями о некоторых ошибках.

При необходимости есть возможность расширить интерфейс приложения, а именно добавить возможность приложение анализировать такую ли фигуру задумывал построить пользователь, добавить расчёт площади 3D фигур и определение их выпуклости.

Стоит отметить что приложение может использоваться на практике в разных сферах деятельности человека, в таких как инженерия, дизайн, архитектура, строительство, проектирование и многие другие. Так как оно позволяет упростить подсчёты и скоротать время как профессионала, так и обычного пользователя. Основная задача на исследование была достигнута.