**`Минобрнауки России**

**Юго-Западный государственный университет**

**Кафедра программной инженерии**

ЛАБОРАТОРНАЯ РАБОТА

по дисциплине «Компьютерная графика»

по теме:

«Программа для формирования изображений»

Выполнили: студенты группы ПО-31б

Василевский В.Я.

Пирогов Д.Ю.

Проверил: старший преподаватель

Ефремов В.В.

Курск 2025 г.

**Вариант – 4**

**Цель работы**– совершенствование базовых навыков обработки растровых изображений с помощью алгоритмических языков программирования.

**Задание:**

1. Написать программу (на языке высокого уровня), реализующую алгоритмы растеризации окружности с их последующей прорисовкой. Координаты центра и радиуса должны задаваться пользователем. Реализовать: - построение окружности по уравнению окружности; - построение окружности по параметрическому уравнению; - алгоритм Брезенхема для построения окружности; - встроенные средства языка программирования.

2. Проиллюстрировать разницу между результатами или её отсутствие.

3. Ответить на контрольные вопросы.

4. Оформить отчёт.

Вариант:

4. Вывести арбеллос.

**Ссылка на репозиторий:**<https://github.com/Vados-rb26/CompGraphLabs>

**Программа на python:**

import math

import tkinter as tk

from tkinter import ttk, messagebox

class CircleRasterizationApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Лабораторная работа №4 - Растеризация окружностей")

self.root.geometry("900x700")

self.image\_width = 600

self.image\_height = 500

self.pixels = [[0 for \_ in range(self.image\_width)] for \_ in range(self.image\_height)]

self.setup\_ui()

def setup\_ui(self):

main\_frame = ttk.Frame(self.root, padding="10")

main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

# Заголовок

title\_label = ttk.Label(main\_frame,

text="Лабораторная работа №4\nРастеризация окружностей",

font=('Arial', 14, 'bold'),

justify=tk.CENTER)

title\_label.grid(row=0, column=0, columnspan=2, pady=(0, 20))

# Левая панель - управление

control\_frame = ttk.LabelFrame(main\_frame, text="Параметры окружности", padding="10")

control\_frame.grid(row=1, column=0, sticky=(tk.W, tk.E, tk.N, tk.S), padx=(0, 10))

# Поля ввода

ttk.Label(control\_frame, text="Центр X:").grid(row=0, column=0, sticky=tk.W, pady=2)

self.center\_x\_entry = ttk.Entry(control\_frame, width=10)

self.center\_x\_entry.insert(0, "300")

self.center\_x\_entry.grid(row=0, column=1, sticky=tk.W, pady=2)

ttk.Label(control\_frame, text="Центр Y:").grid(row=1, column=0, sticky=tk.W, pady=2)

self.center\_y\_entry = ttk.Entry(control\_frame, width=10)

self.center\_y\_entry.insert(0, "250")

self.center\_y\_entry.grid(row=1, column=1, sticky=tk.W, pady=2)

ttk.Label(control\_frame, text="Радиус R:").grid(row=2, column=0, sticky=tk.W, pady=2)

self.radius\_entry = ttk.Entry(control\_frame, width=10)

self.radius\_entry.insert(0, "100")

self.radius\_entry.grid(row=2, column=1, sticky=tk.W, pady=2)

# Кнопки алгоритмов

algorithms\_frame = ttk.LabelFrame(control\_frame, text="Алгоритмы построения", padding="10")

algorithms\_frame.grid(row=3, column=0, columnspan=2, sticky=(tk.W, tk.E), pady=10)

ttk.Button(algorithms\_frame,

text="По уравнению окружности",

command=self.draw\_by\_equation).grid(row=0, column=0, sticky=tk.EW, pady=2)

ttk.Button(algorithms\_frame,

text="По параметрическому уравнению",

command=self.draw\_by\_parametric).grid(row=1, column=0, sticky=tk.EW, pady=2)

ttk.Button(algorithms\_frame,

text="Алгоритм Брезенхема",

command=self.draw\_by\_bresenham).grid(row=2, column=0, sticky=tk.EW, pady=2)

ttk.Button(algorithms\_frame,

text="Встроенные средства",

command=self.draw\_builtin).grid(row=3, column=0, sticky=tk.EW, pady=2)

# Кнопка создания арбелоса

ttk.Button(control\_frame,

text="Создать арбелос (Вариант 4)",

command=self.draw\_arbelos).grid(row=4, column=0, columnspan=2, sticky=tk.EW, pady=5)

# Кнопка очистки

ttk.Button(control\_frame,

text="Очистить",

command=self.clear\_canvas).grid(row=5, column=0, columnspan=2, sticky=tk.EW, pady=2)

# Кнопка сохранения

ttk.Button(control\_frame,

text="Сохранить в PBM",

command=self.save\_to\_pbm).grid(row=6, column=0, columnspan=2, sticky=tk.EW, pady=2)

# Правая панель - холст

canvas\_frame = ttk.LabelFrame(main\_frame, text="Результат", padding="10")

canvas\_frame.grid(row=1, column=1, sticky=(tk.W, tk.E, tk.N, tk.S))

# Холст для рисования

self.canvas = tk.Canvas(canvas\_frame,

width=self.image\_width,

height=self.image\_height,

bg="white",

relief=tk.SUNKEN,

borderwidth=2)

self.canvas.pack(padx=5, pady=5)

# Легенда

legend\_frame = ttk.Frame(canvas\_frame)

legend\_frame.pack(fill=tk.X, pady=5)

ttk.Label(legend\_frame, text="Легенда:", font=('Arial', 9, 'bold')).pack(side=tk.LEFT)

ttk.Label(legend\_frame, text="Уравнение - красный", foreground="red").pack(side=tk.LEFT, padx=10)

ttk.Label(legend\_frame, text="Параметрич. - синий", foreground="blue").pack(side=tk.LEFT, padx=10)

ttk.Label(legend\_frame, text="Брезенхем - зеленый", foreground="green").pack(side=tk.LEFT, padx=10)

# Информационная панель

info\_frame = ttk.LabelFrame(main\_frame, text="Информация", padding="10")

info\_frame.grid(row=2, column=0, columnspan=2, sticky=(tk.W, tk.E), pady=10)

info\_text = """Алгоритмы растеризации окружностей:

1. По уравнению: y = ±√(R² - x²)

2. Параметрический: x = R·cos(t), y = R·sin(t)

3. Алгоритм Брезенхема - наиболее эффективный

4. Арбелос - геометрическая фигура из трех полуокружностей"""

ttk.Label(info\_frame, text=info\_text, justify=tk.LEFT).pack(anchor=tk.W)

def get\_circle\_params(self):

"""Получение параметров окружности из полей ввода"""

try:

cx = int(self.center\_x\_entry.get())

cy = int(self.center\_y\_entry.get())

r = int(self.radius\_entry.get())

return cx, cy, r

except ValueError:

messagebox.showerror("Ошибка", "Введите корректные числовые значения")

return None, None, None

def draw\_pixel(self, x, y, color="black"):

"""Рисование пикселя на холсте"""

if 0 <= x < self.image\_width and 0 <= y < self.image\_height:

self.canvas.create\_rectangle(x, y, x+1, y+1, outline=color, fill=color)

self.pixels[y][x] = 1

def clear\_canvas(self):

"""Очистка холста"""

self.canvas.delete("all")

self.pixels = [[0 for \_ in range(self.image\_width)] for \_ in range(self.image\_height)]

def draw\_by\_equation(self):

"""Построение окружности по уравнению"""

cx, cy, r = self.get\_circle\_params()

if cx is None:

return

for x in range(r + 1):

y = int(math.sqrt(r \* r - x \* x))

# Рисуем во всех октантах

self.draw\_pixel(cx + x, cy + y, "red")

self.draw\_pixel(cx + x, cy - y, "red")

self.draw\_pixel(cx - x, cy + y, "red")

self.draw\_pixel(cx - x, cy - y, "red")

self.draw\_pixel(cx + y, cy + x, "red")

self.draw\_pixel(cx + y, cy - x, "red")

self.draw\_pixel(cx - y, cy + x, "red")

self.draw\_pixel(cx - y, cy - x, "red")

def draw\_by\_parametric(self):

"""Построение окружности по параметрическому уравнению"""

cx, cy, r = self.get\_circle\_params()

if cx is None:

return

steps = int(2 \* math.pi \* r)

for i in range(steps):

t = 2 \* math.pi \* i / steps

x = int(cx + r \* math.cos(t))

y = int(cy + r \* math.sin(t))

self.draw\_pixel(x, y, "blue")

def draw\_by\_bresenham(self):

"""Алгоритм Брезенхема для построения окружности"""

cx, cy, r = self.get\_circle\_params()

if cx is None:

return

x = 0

y = r

delta = 2 - 2 \* r

while y >= 0:

self.draw\_pixel(cx + x, cy + y, "green")

self.draw\_pixel(cx + x, cy - y, "green")

self.draw\_pixel(cx - x, cy + y, "green")

self.draw\_pixel(cx - x, cy - y, "green")

if delta < 0:

d1 = 2 \* (delta + y) - 1

if d1 <= 0:

x += 1

delta += 2 \* x + 1

else:

x += 1

y -= 1

delta += 2 \* (x - y + 1)

elif delta > 0:

d2 = 2 \* (delta - x) - 1

if d2 <= 0:

x += 1

y -= 1

delta += 2 \* (x - y + 1)

else:

y -= 1

delta -= 2 \* y + 1

else:

x += 1

y -= 1

delta += 2 \* (x - y + 1)

def draw\_builtin(self):

"""Построение окружности встроенными средствами"""

cx, cy, r = self.get\_circle\_params()

if cx is None:

return

self.canvas.create\_oval(cx - r, cy - r, cx + r, cy + r, outline="black", width=1)

def draw\_arbelos(self):

"""Построение арбелоса (вариант 4)"""

self.clear\_canvas()

# Параметры арбелоса

base\_y = 400

left\_x = 150

right\_x = 450

center\_x = 300

# Радиусы полуокружностей

big\_radius = 150

left\_radius = 60

right\_radius = 90

# Центры полуокружностей

big\_center = (center\_x, base\_y)

left\_center = (center\_x - big\_radius + left\_radius, base\_y)

right\_center = (center\_x + big\_radius - right\_radius, base\_y)

# Рисуем базовую линию

self.canvas.create\_line(left\_center[0] - left\_radius, base\_y,

right\_center[0] + right\_radius, base\_y,

fill="black", width=2)

# Большая полуокружность (алгоритм Брезенхема)

for x in range(-big\_radius, big\_radius + 1):

y = int(math.sqrt(big\_radius \* big\_radius - x \* x))

if y >= 0:

self.draw\_pixel(big\_center[0] + x, big\_center[1] - y, "green")

# Левая полуокружность (по уравнению)

for x in range(-left\_radius, left\_radius + 1):

y = int(math.sqrt(left\_radius \* left\_radius - x \* x))

if y >= 0:

self.draw\_pixel(left\_center[0] + x, left\_center[1] - y, "red")

# Правая полуокружность (параметрически)

steps = int(2 \* math.pi \* right\_radius)

for i in range(steps // 2):

t = math.pi \* i / (steps // 2)

x = int(right\_center[0] + right\_radius \* math.cos(t))

y = int(right\_center[1] + right\_radius \* math.sin(t))

if y <= right\_center[1]:

self.draw\_pixel(x, y, "blue")

# Подпись

self.canvas.create\_text(300, 50, text="Арбелос (Вариант 4)",

font=('Arial', 12, 'bold'), fill="black")

def save\_to\_pbm(self):

"""Сохранение изображения в PBM формате"""

try:

with open("output.pbm", "w") as f:

f.write("P1\n")

f.write(f"{self.image\_width} {self.image\_height}\n")

for row in self.pixels:

f.write(" ".join(map(str, row)) + "\n")

messagebox.showinfo("Успех", "Изображение сохранено в output.pbm")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось сохранить файл: {str(e)}")

def main():

root = tk.Tk()

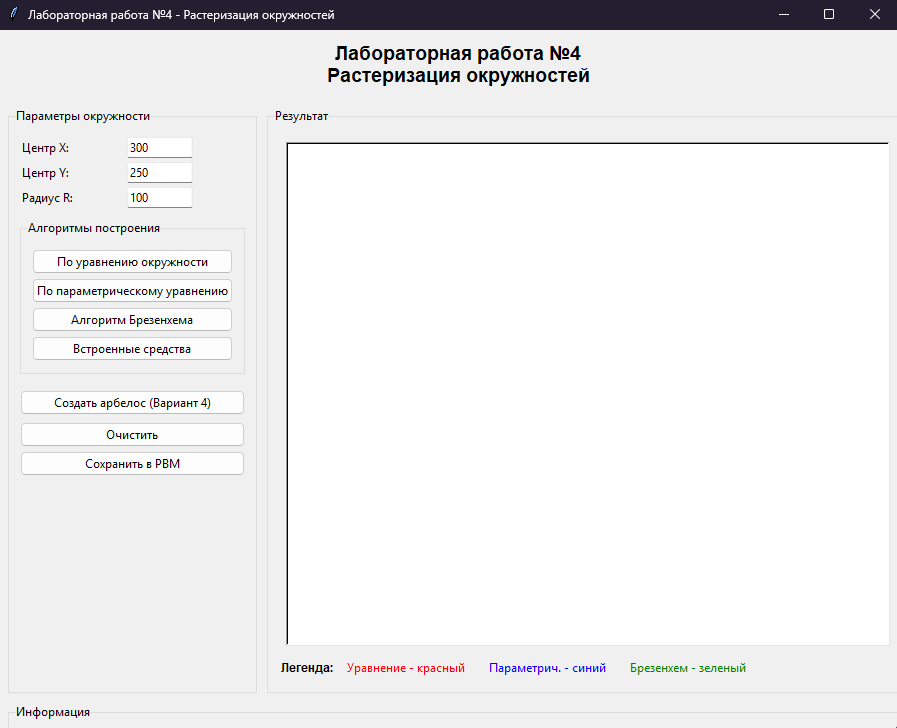
app = CircleRasterizationApp(root)

root.mainloop()

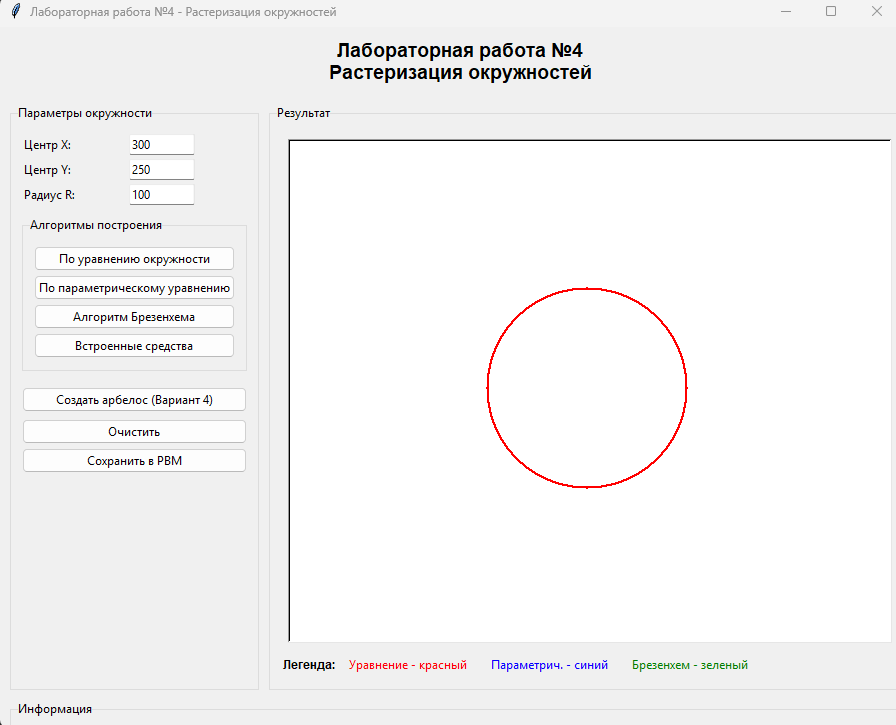
if \_\_name\_\_ == "\_\_main\_\_":

main()

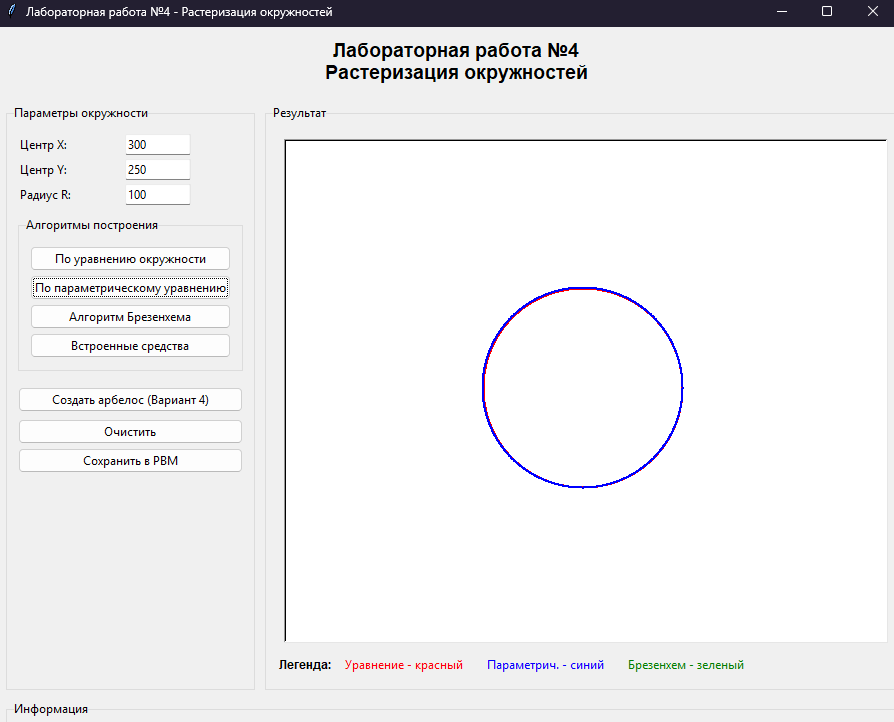
**Порядок работы:  
1.Запуск приложения**

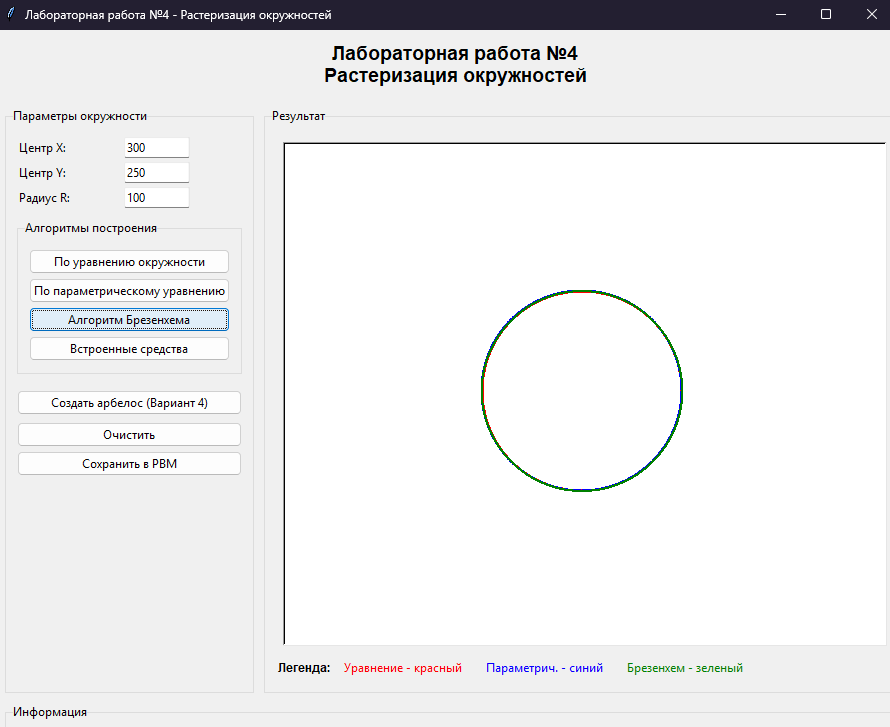


**2.По уравнению окружностей**

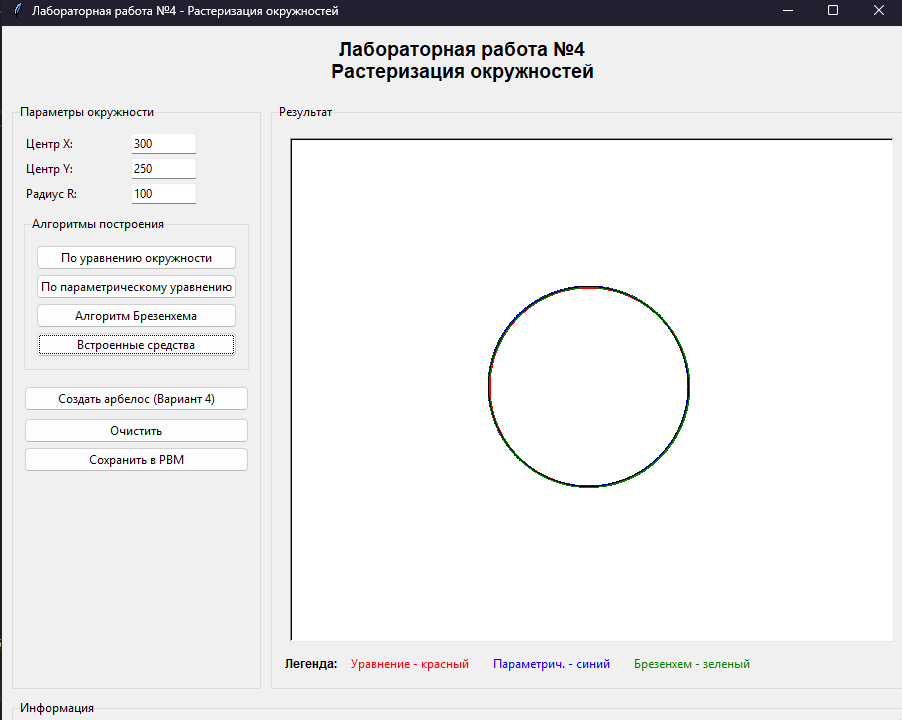


**3.По параметрическому уравнению**

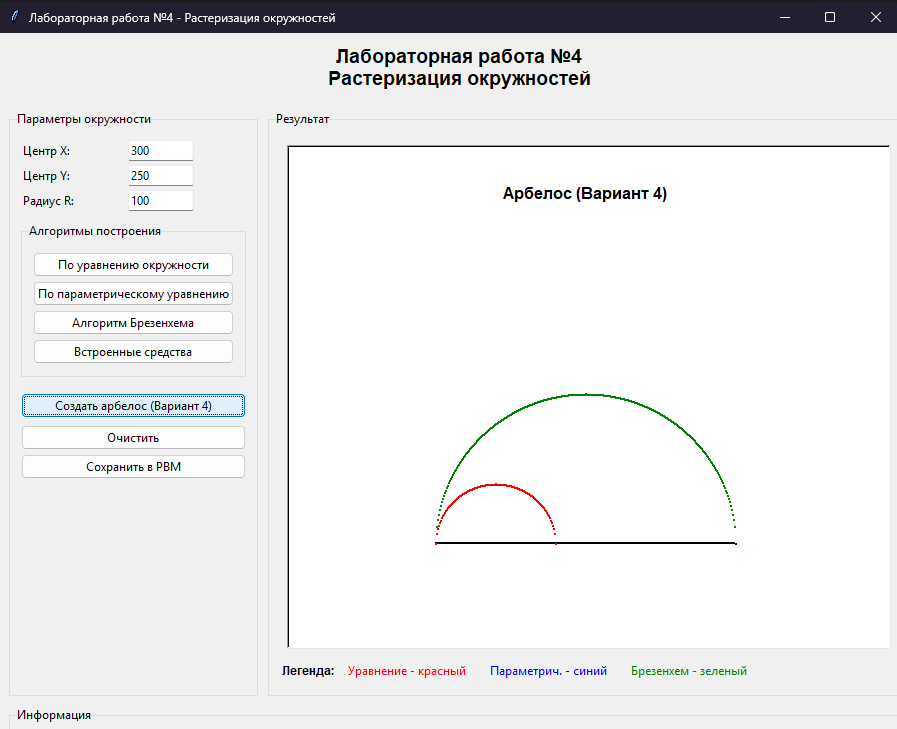


**4.Алгоритм Брезенхерма  
**

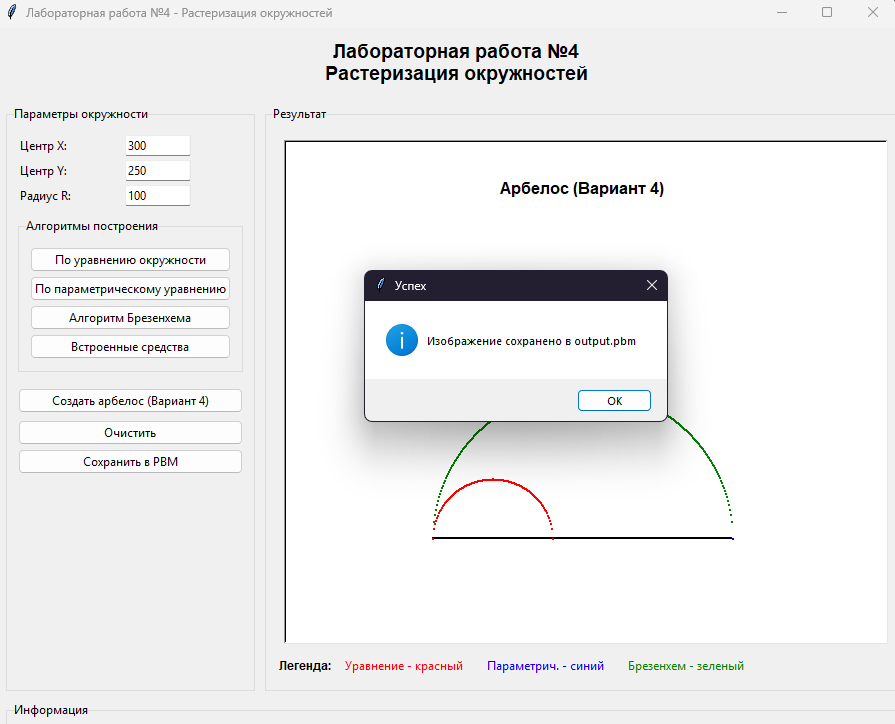
**5.Встроенные средства**

****

**6. Создаем арбелос**

****

**7.Сохраняем результат**

****

**Ответы на контрольные вопросы**

1. Что такое пиксель?

Пиксель (picture element) - минимальный элемент растрового изображения, точка на экране с определенным цветом и координатами. Базовый строительный блок цифровой графики.

2. Что такое растровое изображение?

Изображение, представленное в виде матрицы пикселей (сетки точек). Каждый пиксель хранит информацию о цвете. Примеры: JPEG, PNG, BMP, PBM.

3. Для чего нужны алгоритмы растеризации окружностей?

Для преобразования математического описания окружности (уравнение, параметры) в набор пикселей на растровом дисплее или принтере. Используются в компьютерной графике, САПР, играх.

4. Какие ещё алгоритмы растеризации окружностей существуют?

Помимо рассмотренных в работе алгоритмов (по уравнению окружности, параметрического и алгоритма Брезенхема), существует несколько других подходов к растеризации окружностей. Одним из наиболее известных является алгоритм средней точки (Midpoint Circle Algorithm), который представляет собой модификацию алгоритма Брезенхема и также основан на принятии решений о выборе следующего пикселя на основе значения функции в средней точке между кандидатами. Также существуют алгоритмы, использующие исключительно целочисленную арифметику для повышения производительности, и антиалиасинговые алгоритмы, которые применяют сглаживание для уменьшения ступенчатости контуров за счёт использования нескольких оттенков цвета. Некоторые специализированные алгоритмы оптимизированы для конкретного оборудования или используют предварительно вычисленные таблицы для увеличения скорости работы.