МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по лабораторной работе №7 по курсу**

**«МЕТОДЫ ПОИСКОВОЙ ОПТИМИЗАЦИИ»**

Работу выполнили

Студенты 46 группы

Нагалевский А.М.

Прозоров М.С.

Преподаватель:

Нигодин Е.А.

Краснодар 2023

**Цель работы:** разработать алгоритм бактериальной оптимизации на определенной функции.

**Ход работы:**

Одним из самых популярных методов оптимизации, вдохновленные природными процессами, является алгоритм "бактериальной оптимизации" (Bacterial Foraging Optimization, BFO). Этот метод оптимизации был предложен индийскими учеными в 2002 году и базируется на моделировании поведения бактерий в процессе питания.

Рассмотрим общий принцип работы алгоритма BFO:

1. **Инициализация популяции бактерий**: Начальное положение каждой "бактерии" представляет собой точку в пространстве поиска решений.
2. **Оценка функции приспособленности:** Каждая бактерия оценивает свою приспособленность в соответствии с задачей оптимизации. Это значение определяется функцией приспособленности, которую нужно оптимизировать.
3. **Движение бактерий:** Бактерии перемещаются в пространстве поиска в соответствии с правилами, моделирующими их поведение при поиске пищи. В процессе движения они могут оставлять "химические следы" (отпечатки), которые привлекают или отталкивают других бактерий.
4. **Обновление химических следов:** Химические следы обновляются в зависимости от успеха бактерий в поиске лучших решений.
5. **Итерации**: Процесс оценки, движения и обновления химических следов повторяется на протяжении нескольких итераций до тех пор, пока не будет достигнут критерий остановки.
6. **Выбор оптимального решения**: По завершении алгоритма выбирается лучшее решение, найденное бактериальной оптимизацией.

Теперь рассмотрим достоинства и недостатки данного алгоритма.

Плюсы алгоритма бактериальной оптимизации (BFO):

* **Простота реализации**: Алгоритм BFO относительно прост в понимании и реализации. Это может быть преимуществом для быстрого применения в решении оптимизационных задач.
* **Работа с большим числом переменных:** BFO может эффективно работать с задачами, включающими большое число переменных, что может быть проблемой для некоторых других методов оптимизации.
* **Относительная устойчивость к локальным оптимумам**: Механизм обновления химических следов может помочь избегать застревания в локальных оптимумах, что является преимуществом при работе с многомодальными функциями.
* **Применимость к различным задачам**: BFO может применяться для разнообразных задач оптимизации и адаптироваться к различным типам функций приспособленности.

Минусы алгоритма бактериальной оптимизации (BFO):

* **Не гарантируется глобальная оптимальность**: Как и многие эволюционные методы оптимизации, BFO не гарантирует нахождение глобального оптимума. Его эффективность может зависеть от природы конкретной задачи.
* **Чувствительность к параметрам**: Работа BFO может зависеть от выбора параметров, таких как размер популяции бактерий, скорость движения и т.д. Неправильный выбор параметров может снизить эффективность алгоритма.
* **Не всегда быстрый поиск**: BFO может быть менее эффективным в сравнении с некоторыми современными алгоритмами оптимизации, основанными на более сложных и эффективных стратегиях.
* **Возможная сходимость к локальным оптимумам**: Несмотря на механизм обновления химических следов, BFO может все равно сталкиваться с проблемой сходимости к локальным оптимумам в некоторых сложных функциях.
* **Низкая скорость сходимости**: В сравнении с некоторыми современными методами оптимизации, BFO может иметь низкую скорость сходимости, что означает большее количество итераций для достижения оптимального решения.

Для создания программы используется язык программирования Python 3.11 и среда разработки PyCharm. Для графической визуализации используется графический фреймворк Tkinter и Matplotlib.

В созданной программе одно главное активное окно.

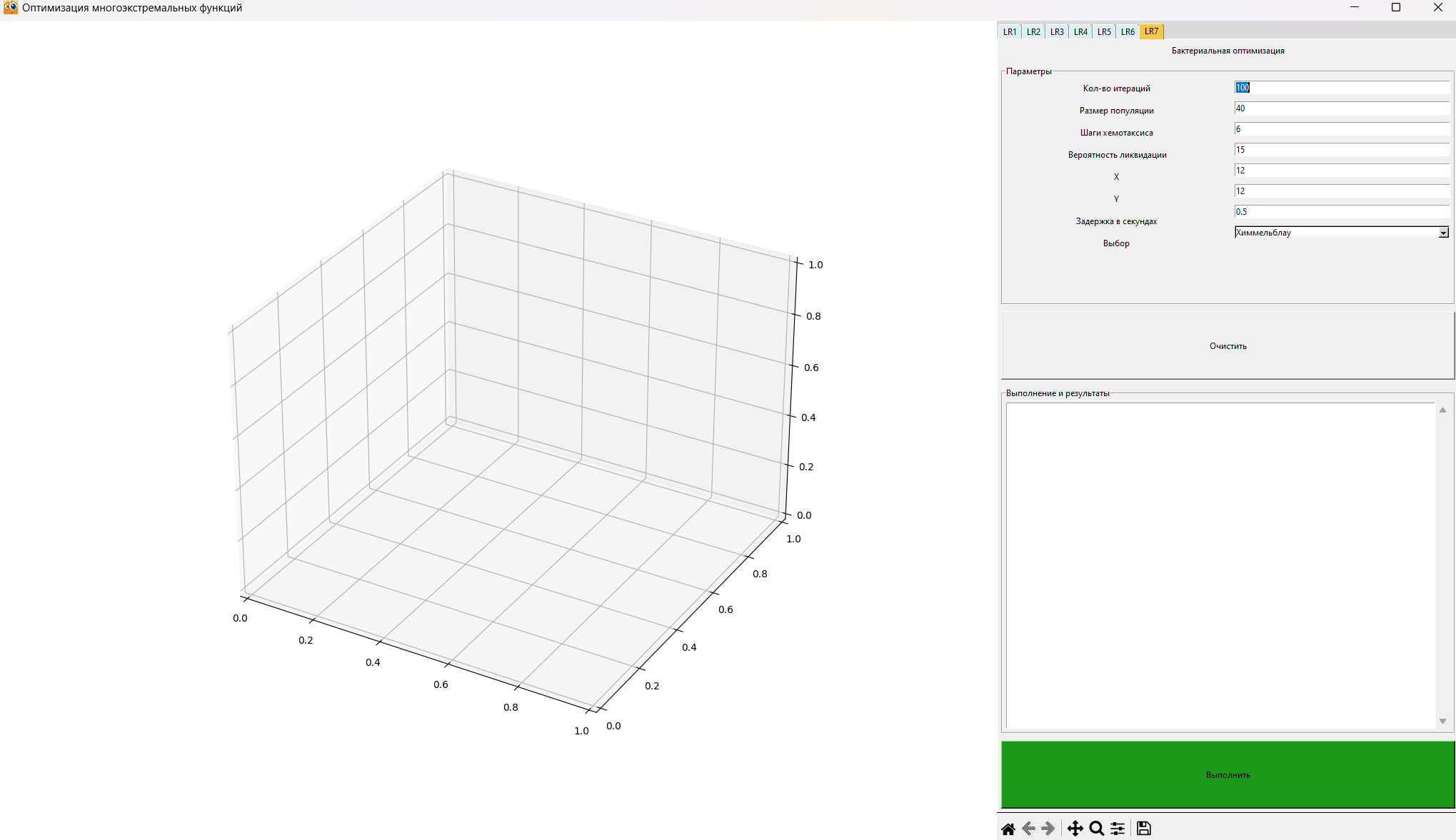


Рисунок 1 – Главное окно программы.

В данном окне можно выбрать необходимую лабораторную работу, в данном случае «7», ввести количество итераций алгоритма, размер популяции, шаги хетомаксиса, вероятность ликвидации и задержку в секундах.

В поле «Выполнение и результаты» динамически выводятся результаты работы алгоритма в виде шагов, представленных координатами и значениями оптимизируемой функции в этих координатах. На рисунке 2 показано поле «Выполнение и результаты» до запуска программы.

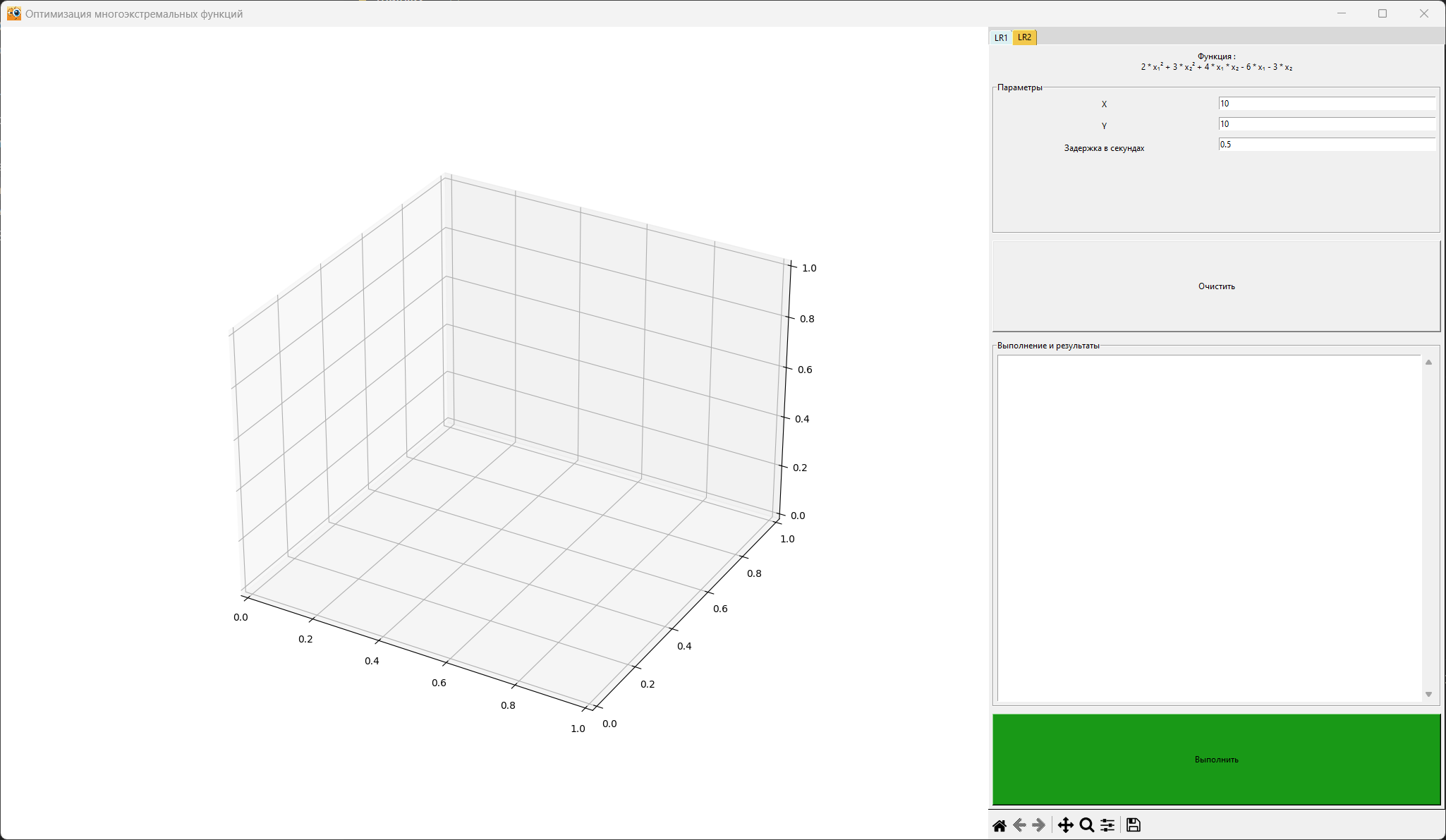


Рисунок 2 – Выполнение и результаты.

При нажатии на кнопку “Выполнить” на главном окне программы отображается необходимая нам функция, как показано на рисунке 4. Причем, более высокие значения функции показана ярко желтым цветом, а самые низкие темно серым цветом.

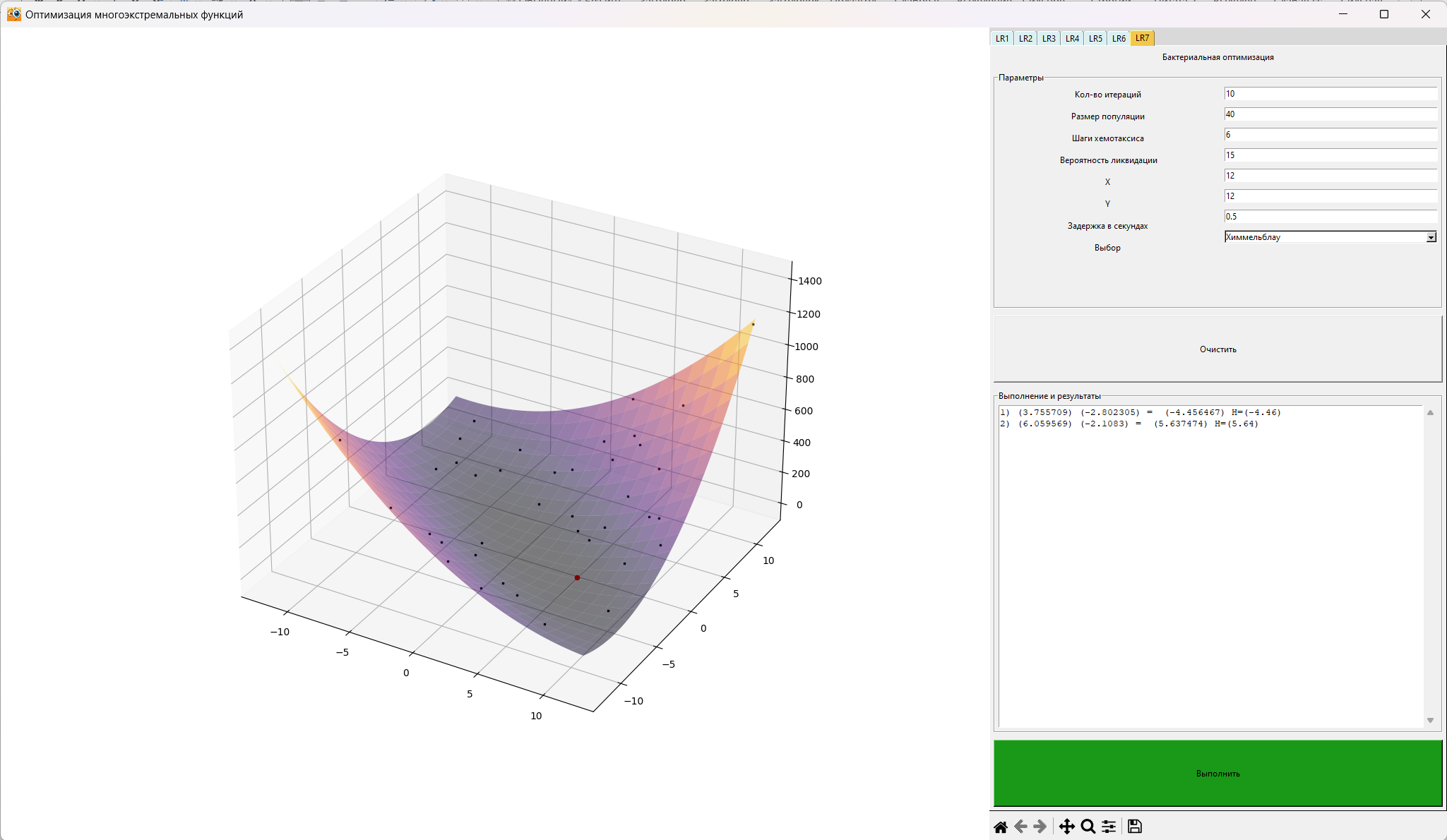


Рисунок 4 – Функция Химмельблау.

Для запуска алгоритма необходимо нажать кнопку «Выполнить», при этом поле «Выполнение и результаты» начнёт динамически заполнятся, а на отображаемой функции можно наглядно увидеть функционирование алгоритма - искомые точки, которые также отображаются динамически как показано на рисунке 5. При полном выполнении программа выводит окно, уведомляющие нас об окончании работы алгоритма.

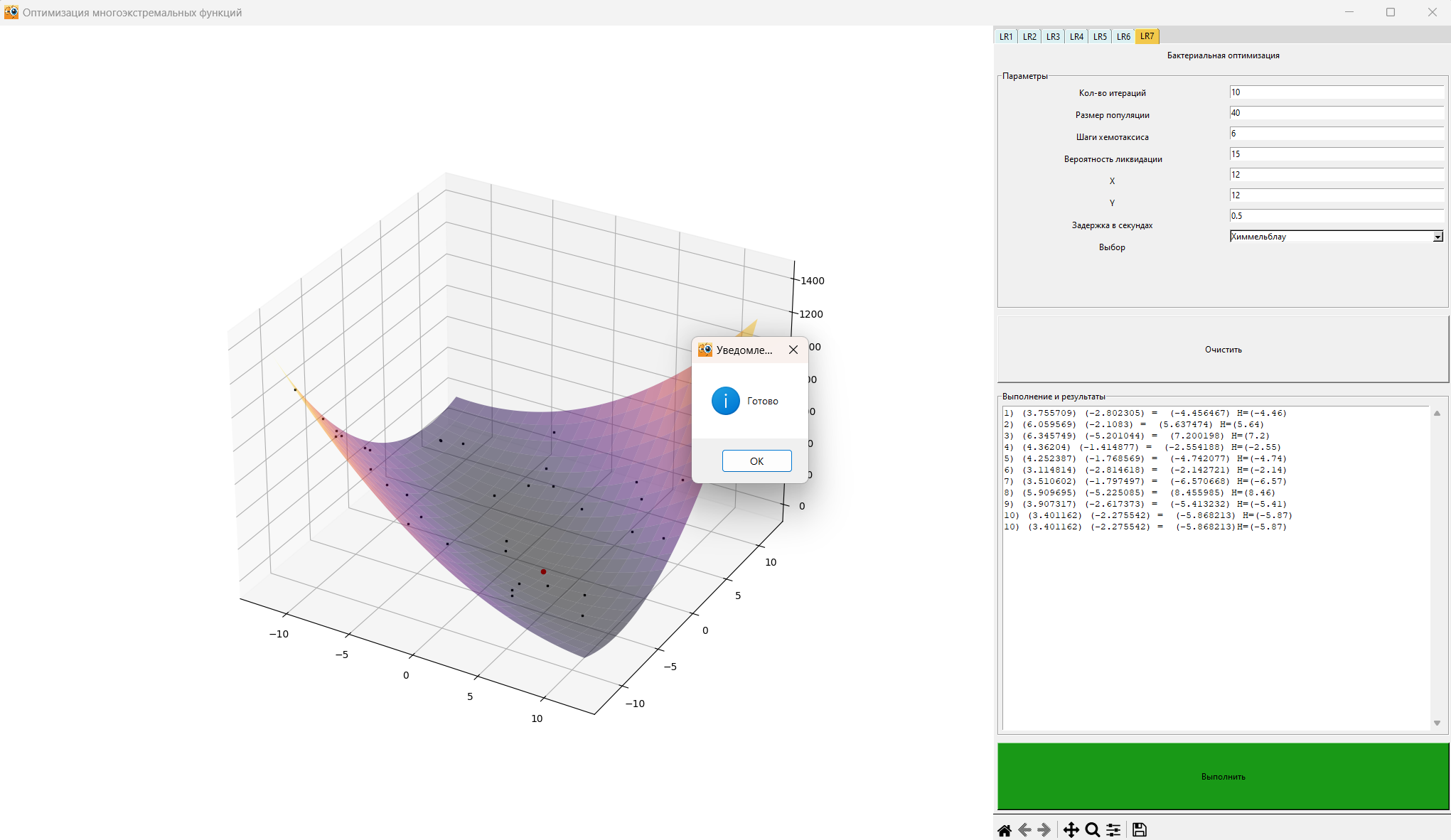


Рисунок 5 – Результат работы программы.

**Вывод:** в ходе работы был изучен и реализован алгоритм бактериальной оптимизации на определенной функции.

**Листинг программы:**

**Файл bacterias.py**

import random

from operator import itemgetter

class Bacteria:

def \_\_init\_\_(self, func, population, chemotaxis, licvidation, position\_x, position\_y):

self.func = func

self.pos\_x = float(position\_x)

self.pos\_y = float(position\_y)

self.pop\_numb = int(population)

self.agents = [[random.uniform(-self.pos\_x, self.pos\_x), random.uniform(-self.pos\_y, self.pos\_y), 0.0, 0.0] for

\_ in

range(self.pop\_numb)]

for i in self.agents:

i[2] = self.func(i[0], i[1])

i[3] = i[2]

self.chemo\_step = chemotaxis

self.licvid = licvidation

def chemotaxis(self, coef):

for bac in self.agents:

vec = [coef \* random.uniform(-1, 1), coef \* random.uniform(-1, 1)]

for \_ in range(self.chemo\_step):

f = bac[2]

bac[0] += vec[0] # X

bac[1] += vec[1] # Y

bac[2] = self.func(bac[0], bac[1]) # Z / Фитнес-Функция

bac[3] += bac[2] # Health

if f < bac[2]:

vec = [coef \* random.uniform(-1, 1), coef \* random.uniform(-1, 1)]

def reproduction(self):

self.agents = sorted(self.agents, key=itemgetter(3), reverse=False)

for i in range(self.pop\_numb // 2):

self.agents[self.pop\_numb // 2 + i] = self.agents[i].copy()

def elimnination(self):

for bac in self.agents:

if random.uniform(0, 1) <= self.licvid:

bac[0] = random.uniform(-self.pos\_x, self.pos\_x)

bac[1] = random.uniform(-self.pos\_y, self.pos\_y)

bac[2] = self.func(bac[0], bac[1])

bac[3] = bac[2]

def get\_best(self):

return sorted(self.agents, key=itemgetter(2), reverse=False)[0]

**Файл main.py**

import tkinter

import time

import sys

from tkinter import \*

from tkinter import scrolledtext, messagebox

from tkinter.ttk import Combobox, Notebook, Style

from matplotlib import pyplot as plt

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

from Gradient import make\_data\_lab\_1, funct\_consider

from SLSQP import make\_data\_lab\_2, kp

from Rosenbrock\_function import make\_data\_lab\_3

from genetic\_algorithm\_l3 import GeneticAlgorithmL3

from pso import PSO

from bees import Bees

from immune import Immunity

from bacterias import Bacteria

from functions import \*

def main():

window = Tk()

window.iconbitmap(r'pic/hto.ico')

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

window.geometry("%dx%d" % (width, height))

window.title("Оптимизация многоэкстремальных функций")

fig = plt.figure(figsize=(14, 14))

fig.add\_subplot(projection='3d')

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tkinter.LEFT, fill=tkinter.BOTH)

toolbar = NavigationToolbar2Tk(canvas, window)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tkinter.LEFT, fill=tkinter.BOTH)

sky = "#DCF0F2"

yellow = "#F2C84B"

style = Style()

style.theme\_create("dummy", parent="alt", settings={

"TNotebook": {"configure": {"tabmargins": [2, 5, 2, 0]}},

"TNotebook.Tab": {

"configure": {"padding": [5, 1], "background": sky},

"map": {"background": [("selected", yellow)],

"expand": [("selected", [1, 1, 1, 0])]}}})

style.theme\_use("dummy")

tab\_control = Notebook(window)

def draw\_lab\_7():

fig.clf()

iter\_number = int(txt\_1\_tab\_7.get())

population = int(txt\_2\_tab\_7.get())

xemotaxis = int(txt\_3\_tab\_7.get())

licvid = float(txt\_4\_tab\_7.get())

pos\_x = int(txt\_5\_tab\_7.get())

pos\_y = int(txt\_6\_tab\_7.get())

delay = txt\_7\_tab\_7.get()

if combo\_tab\_7.get() == "Химмельблау":

func = himmelblau\_2

x, y, z = make\_data\_himmelblau(pos\_x, pos\_y)

elif combo\_tab\_7.get() == "Розенброка":

func = rosenbrock\_2

x, y, z = make\_data\_rosenbrock(pos\_x, pos\_y)

else:

func = rastrigin\_2

x, y, z = make\_data\_rastrigin(pos\_x, pos\_y)

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

bacterias = Bacteria(func, population, xemotaxis, licvid, pos\_x, pos\_y)

for i in range(iter\_number):

bacterias.chemotaxis(1 / (i + 1))

bacterias.reproduction()

bacterias.elimnination()

for bac in bacterias.agents:

ax.scatter(bac[0], bac[1], bac[2], c="black", s=1, marker="s")

b = bacterias.get\_best()

ax.scatter(b[0], b[1], b[2], c="red")

txt\_tab\_7.insert(INSERT,

f"{i + 1}) ({round(b[0], 6)})"

f" ({round(b[1], 6)}) = "

f" ({round(b[2], 6)})"

f" H=({round(b[3], 2)})\n")

canvas.draw()

window.update()

time.sleep(float(delay))

fig.clf()

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

for bac in bacterias.agents:

ax.scatter(bac[0], bac[1], bac[2], c="black", s=1, marker="s")

b = bacterias.get\_best()

ax.scatter(b[0], b[1], b[2], c="red")

txt\_tab\_7.insert(INSERT,

f"{i + 1}) ({round(b[0], 6)})"

f" ({round(b[1], 6)}) = "

f" ({round(b[2], 6)})"

f"H=({round(b[3], 2)})\n")

canvas.draw()

window.update()

canvas.draw()

window.update()

messagebox.showinfo('Уведомление', 'Готово')

def delete\_lab\_7():

txt\_tab\_7.delete(1.0, END)

tab\_7 = Frame(tab\_control)

tab\_control.add(tab\_7, text="LR7")

main\_f\_tab\_7 = LabelFrame(tab\_7, text="Параметры")

left\_f\_tab\_7 = Frame(main\_f\_tab\_7)

right\_f\_tab\_7 = Frame(main\_f\_tab\_7)

txt\_f\_tab\_7 = LabelFrame(tab\_7, text="Выполнение и результаты")

lbl\_1\_tab\_7 = Label(left\_f\_tab\_7, text="Кол-во итераций")

lbl\_2\_tab\_7 = Label(left\_f\_tab\_7, text="Размер популяции")

lbl\_3\_tab\_7 = Label(left\_f\_tab\_7, text="Шаги хемотаксиса")

lbl\_4\_tab\_7 = Label(left\_f\_tab\_7, text=" Вероятность ликвидации")

lbl\_5\_tab\_7 = Label(left\_f\_tab\_7, text="Задержка в секундах")

lbl\_6\_tab\_7 = Label(tab\_7, text="Бактериальная оптимизация")

lbl\_7\_tab\_7 = Label(left\_f\_tab\_7, text="X")

lbl\_8\_tab\_7 = Label(left\_f\_tab\_7, text="Y")

lbl\_9\_tab\_7 = Label(left\_f\_tab\_7, text="Выбор")

txt\_1\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_1\_tab\_7.insert(0, "100")

txt\_2\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_2\_tab\_7.insert(0, "40")

txt\_3\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_3\_tab\_7.insert(0, "6")

txt\_4\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_4\_tab\_7.insert(0, "15")

txt\_5\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_5\_tab\_7.insert(0, "12")

txt\_6\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_6\_tab\_7.insert(0, "12")

txt\_7\_tab\_7 = Entry(right\_f\_tab\_7)

txt\_7\_tab\_7.insert(0, "0.5")

combo\_tab\_7 = Combobox(right\_f\_tab\_7)

combo\_tab\_7['values'] = ("Химмельблау", "Розенброка")

combo\_tab\_7.set("Химмельблау")

txt\_tab\_7 = scrolledtext.ScrolledText(txt\_f\_tab\_7)

btn\_del\_tab\_7 = Button(tab\_7, text="Очистить", command=delete\_lab\_7)

btn\_tab\_7 = Button(tab\_7, text="Выполнить", foreground="black", background="#199917", command=draw\_lab\_7)

lbl\_6\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

main\_f\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_7.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_7.pack(side=RIGHT, fill=BOTH, expand=True)

lbl\_1\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_2\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_3\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_4\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_7\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_8\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_5\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_9\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_1\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_2\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_3\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_4\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_5\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_6\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_7\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

combo\_tab\_7.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_tab\_7.pack(padx=5, pady=5, fill=BOTH, expand=True)

btn\_tab\_7.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_7.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_7.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

tab\_control.pack(side=RIGHT, fill=BOTH, expand=True)

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()