**4. praktiskais darbs 2.semestris**

Es apmeklēju 01.03.2023. praktisko nodarbību, tāpēc 5. uzdevums jau tika izpildīts nodarbības laikā.

**1. uzdevums**

Sastādīt programmu, kas nosaka punkta koordinātu XX un YY vērtības

Diagram

Description automatically generated

ja riņķa rādiusu R un punkta koordinātas X un Y ievada lietotājs no tastatūras.

**Kods:**

# Programmas nosaukums: punkta koordinātu XX un YY vērtības

# 1. uzdevums (1MPR04\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas nosaka punkta koordinātu XX un YY vērtības, ja riņķa rādiusu R un punkta koordinātas X un Y ievada lietotājs no tastatūras.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

def is\_real\_positive(n):

# Pārbauda vai simbolu virkne ir reāla pozitīva vērtība

# Ja ir tad return True

# Ja nav tad return False

# vai n >= 0

# n - simbolu virkne (str)

try:

n = float(n)

b = math.sqrt(n)

c = 1 / n

except:

return False

else:

return True

def is\_real\_number\_velreiz(X): # Bezgalīgi daudz reizes ievāda

# Prasa ievadīt "X" skaitli un pārbauda, vai tas ir reāls skaitlis vai nav

# Ja tas ir reāls skaitlis tad atdot float(x1) ievadīto

# X - ievada ka str vērtību, atspoguļojas tikai kosmetiski (Ievaditet "X" vērtību)

x1 = input("Ievadiet " + X + " ===> ")

while True:

try:

x1 = float(x1)

except:

x1 = input("Kļūda! " + X + " ir jābut skaitlim!\nIevadiet " + X + " ===> ")

else:

return float(x1)

def is\_real\_not\_negative(n):

# Pārbauda vai simbolu virkne n ir reāls nenegatīvs skaitlis vai nē

# Ja ir, tad return True

# Ja nē, tad return False

try:

n = float(n)

b = math.sqrt(n)

except:

return False

else:

return True

def krustpunkts(punkts, radius):

# Atrod nepieciešāmo krustpunktu un return kā kortežu (xx, yy)

# punkts - (x1,y1) kortežs

# radius - rādiusa vērtība [0; + bezg)

(x, y) = punkts

if (x, y) == (0, 0) and radius == 0:

return False # Dalīšana ar 0

d = math.sqrt(x \* x + y \* y)

xx = x \* radius / d

yy = y \* radius / d

return (xx, yy)

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

x1 = is\_real\_number\_velreiz("X")

y1 = is\_real\_number\_velreiz("Y") # float(input("Ievadiet Y ===> "))

p1 = (x1, y1)

radius = is\_real\_number\_velreiz("R")# input("Ievadiet rādiusu ===> ")

while not is\_real\_not\_negative(radius):

radius = float(input("Kļūda! Rādiusam ir jābut reālai nenegatīvai vērtībai!\nIevadiet rādiusu ===> "))

radius = float(radius)

if krustpunkts(p1, radius) == False:

print("Šajā gadījumā nevar viennozīmīgi noteikt atbildi.") #if p1 == (0, 0) and radius == 0:

else:

print("(XX,YY) = " + str(krustpunkts(p1, radius)))

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Graphical user interface, text

Description automatically generated

3)

Text

Description automatically generated

4)

A picture containing text

Description automatically generated

5)

Text

Description automatically generated

6)

Text

Description automatically generated

**2. uzdevums**

Sastādīt programmu, kas noskaidro vai šīs taisnes krustojas tā, lai izveidotos trijstūris. Koeficientus ievada no tastatūras. Ja taisnes krustojoties veido trijstūri, tad aprēķināt tā laukumu.

*Izvadāmo rezultātu piemēri:*

• Taisnes novietotas tā, ka neveidojas trijstūris

• Taisnes krustojas tā, ka veidojas trijstūris, kura laukums ir 5.3

**Kods:**

# Programmas nosaukums: Noskaidro vai šīs taisnes krustojas tā, lai izveidotos trijstūris.

# 2. uzdevums (1MPR04\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas noskaidro vai šīs taisnes krustojas tā, lai izveidotos trijstūris. Koeficientus a\_{11},\ a\_{12},\ a\_{13},\ a\_{21},\ a\_{22},\ a\_{23},\ a\_{31},\ a\_{32},\ a\_{33} ievada no tastatūras. Ja taisnes krustojoties veido trijstūri, tad aprēķināt tā laukumu.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

def det(a, b, c, d):

# Atgriež determinanta vērtību

# a - skaitlis float

# b - skaitlis float

# c - skaitlis float

# d - skaitlis float

return a \* d - b \* c

def mala(kp1, kp2): # kp1 = (a,b) kp2 = (c,d)

# Malas garums pēc diviem punktiem

# Atgriež malas garumu

# kp1 - 1. punkts

# kp2 - 2. punkts

return math.sqrt((kp1[0] - kp2[0]) \* (kp1[0] - kp2[0]) + (kp1[1] - kp2[1]) \* (kp1[1] - kp2[1]))

def laukums(kp1, kp2, kp3): # kp1 =(a,b) kp2 = (c,d) kp3 = (f,g)

# Noskaidro trījstūra laukumu ar Herona formulu pēc trīs krustpunktiem

# kp1 - 1. krustpunkts

# kp2 - 2. krustpunkts

# kp3 - 3. krustpunkts

m1 = mala(kp1, kp2)

m2 = mala(kp1, kp3)

m3 = mala(kp2, kp3)

p = (m1 + m2 + m3) / 2

return math.sqrt(p \* (p - m1) \* (p - m2) \* (p - m3))

def vai\_pa\_pariem\_krustojas(t1, t2, t3):

# Funkcija noskaidro vai pa pariem krustojas trīs taisnes

# return True, ja krustojas

# return False, ja nekrustojas

# t1 - pirmā taisne

# t2 - otrā taisne

# t3 - treša taisne

if (det(t1[0], t1[1], t2[0], t2[1]) \* det(t1[0], t1[1], t3[0], t3[1]) \* det(t2[0], t2[1], t3[0], t3[1])) == 0:

return False

else:

return True

def krustpunkts(t1, t2):

# t1 - pirmā taisne

# t2 - otrā taisne

# return krustpunktu divām taisnēm (x,y)

d = det(t1[0], t1[1], t2[0], t2[1])

dx = det(t1[1], t1[2], t2[1], t2[2])

dy = det(t1[2], t1[0], t2[2], t2[0])

x = dx / d

y = dy / d

return (x, y)

def is\_real\_number\_velreiz(X): # Bezgalīgi daudz reizes ievāda

# Prasa ievadīt "X" skaitli un pārbauda, vai tas ir reāls skaitlis vai nav

# Ja tas ir reāls skaitlis tad atdot float(x1) ievadīto

# X - ievada ka str vērtību, atspoguļojas tikai kosmetiski (Ievaditet "X" vērtību)

x1 = input("Ievadiet " + X + " ===> ")

while True:

try:

x1 = float(x1)

except:

x1 = input("Kļūda! " + X + " ir jābut skaitlim!\nIevadiet " + X + " ===> ")

else:

return float(x1)

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

a11 = is\_real\_number\_velreiz("a11")

a12 = is\_real\_number\_velreiz("a12")

a13 = is\_real\_number\_velreiz("a13")

t1 = (a11, a12, a13)

a21 = is\_real\_number\_velreiz("a21")

a22 = is\_real\_number\_velreiz("a22")

a23 = is\_real\_number\_velreiz("a23")

t2 = (a21, a22, a23)

a31 = is\_real\_number\_velreiz("a31")

a32 = is\_real\_number\_velreiz("a32")

a33 = is\_real\_number\_velreiz("a33")

t3 = (a31, a32, a33)

if vai\_pa\_pariem\_krustojas(t1, t2, t3):

kp1 = krustpunkts(t1, t2)

kp2 = krustpunkts(t1, t3)

kp3 = krustpunkts(t2, t3)

s = laukums(kp1, kp2, kp3)

if s != 0:

print("Taisnes krustojas, ierobežojot laukumu S = " + str(s))

else:

print("Taisnes krustojas vienā punktā. Trijsturūris neveidojas.")

else:

print("Taisnes nekrustojas")

**Testa piemēri:**

1)

A picture containing text, keyboard

Description automatically generated

2)

Text

Description automatically generated

3)

A screenshot of a computer

Description automatically generated with medium confidence

4)

A screenshot of a computer

Description automatically generated with medium confidence

5)

Text

Description automatically generated

6)

Text

Description automatically generated

**3. uzdevums**

Sastādīt programmu, kas realizē četras aritmētiskās darbības ar daļskaitļiem. Piespiežot pogu =, tiek parādīts rezultāts atbilstoši, tai darbības zīmei, kas redzama uz pogas. Darbības zīme uz pogas tiek mainīta to piespiežot vienu vai vairākas reizes, līdz iegūst vajadzīgo zīmi. Daļu saucēji un skaitītāji ir naturāli skaitļi, bet, ja izpildītās darbības rezultātā ir iegūts negatīvs skaitlis, tad zīme tiek parādīta pirms daļsvītras. Veikt ievaddatu korektuma pārbaudi to ievadīšanas brīdī

Shape, rectangle

Description automatically generated

**Kods:**

# Programmas nosaukums: Četras aritmētiskās darbības ar daļskaitļiem.

# 3. uzdevums (1MPR04\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē četras aritmētiskās darbības ar daļskaitļiem. Piespiežot pogu =, tiek parbaudeādīts rezultāts atbilstoši, tai darbības zīmei, kas redzama uz pogas.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

import tkinter # Importējam tkinter moduli

def mainit\_zimi():

cetras\_zimes\_mainisana\_plus\_minus\_reizinat\_dalit(button)

def cetras\_zimes\_mainisana\_plus\_minus\_reizinat\_dalit(ee):

index = zimes\_veidi.index(ee['text'])

ee.config(text=zimes\_veidi[(index + 1) % 4])

def saskaitit(dala\_1, dala\_2):

saucejs = dala\_1[1] \* dala\_2[1]

skaititajs = dala\_1[0] \* dala\_2[1] + dala\_1[1] \* dala\_2[0]

return (skaititajs, saucejs)

def atnemt(dala\_1, dala\_2):

saucejs = dala\_1[1] \* dala\_2[1]

skaititajs = dala\_1[0] \* dala\_2[1] - dala\_1[1] \* dala\_2[0]

return (skaititajs, saucejs)

def reizinat(dala\_1, dala\_2):

skaititajs = dala\_1[0] \* dala\_2[0]

saucejs = dala\_1[1] \* dala\_2[1]

return (skaititajs, saucejs)

def dalit(dala\_1, dala\_2):

skaititajs = dala\_1[0] \* dala\_2[1]

saucejs = dala\_1[1] \* dala\_2[0]

return (skaititajs, saucejs)

def gcd(a, b):

while b != 0:

c = a % b

a = b

b = c

return a

def result():

label\_skaititajs\_result.config(text="")

label\_saucejs\_result.config(text="")

label\_minus.config(text="")

dala\_1 = (int(input\_skaititajs\_1.get()), int(input\_saucejs\_1.get()))

dala\_2 = (int(input\_skaititajs\_2.get()), int(input\_saucejs\_2.get()))

index = zimes\_veidi.index(button['text'])

if dala\_1[0] > 0 and dala\_2[0] > 0 and dala\_1[1] > 0 and dala\_2[1] > 0:

minus = ""

if index == 0:

finals = saskaitit(dala\_1, dala\_2)

elif index == 1:

finals = atnemt(dala\_1, dala\_2)

if finals[0] < 0:

finals = (abs(finals[0]), finals[1])

minus = "-"

elif index == 2:

finals = reizinat(dala\_1, dala\_2)

else:

finals = dalit(dala\_1, dala\_2)

gcdala\_1 = gcd(finals[0], finals[1])

label\_skaititajs\_result.config(text=str(finals[0] // gcdala\_1))

label\_saucejs\_result.config(text=str(finals[1] // gcdala\_1))

label\_minus.config(text=minus)

def is\_int\_skaititajs\_1(event):

global parbaude\_skaititajs\_1

parbaude = input\_skaititajs\_1.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

parbaude = 1 / parbaude

except:

if input\_skaititajs\_1.get() == "": # ja nekas nav ierakstīts

input\_skaititajs\_1.config(bg="white") # tad iekrasosīm baltā

submit\_button['state'] = tkinter.DISABLED

parbaude\_skaititajs\_1 = False # tad globalais parbaude\_skaititajs\_1 ir False

else:

input\_skaititajs\_1.config(bg="red")

submit\_button['state'] = tkinter.DISABLED

parbaude\_skaititajs\_1 = False

else:

input\_skaititajs\_1.config(bg="white")

parbaude\_skaititajs\_1 = True

check() # Pārbaudam vai visiem entry ir ierakstīti reāli skaitli

def is\_int\_skaititajs\_2(event):

global parbaude\_skaititajs\_2

parbaude = input\_skaititajs\_2.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

parbaude = 1 / parbaude

except:

if input\_skaititajs\_2.get() == "":

input\_skaititajs\_2.config(bg="white")

submit\_button['state'] = tkinter.DISABLED

parbaude\_skaititajs\_2 = False

else:

input\_skaititajs\_2.config(bg="red")

submit\_button['state'] = tkinter.DISABLED

parbaude\_skaititajs\_2 = False

else:

input\_skaititajs\_2.config(bg="white")

parbaude\_skaititajs\_2 = True

check()

def is\_int\_saucejs\_1(event):

global parbaude\_saucejs\_1

parbaude = input\_saucejs\_1.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

parbaude = 1 / parbaude

except:

if input\_saucejs\_1.get() == "":

input\_saucejs\_1.config(bg="white")

submit\_button['state'] = tkinter.DISABLED

parbaude\_saucejs\_1 = False

else:

input\_saucejs\_1.config(bg="red")

submit\_button['state'] = tkinter.DISABLED

parbaude\_saucejs\_1 = False

else:

input\_saucejs\_1.config(bg="white")

parbaude\_saucejs\_1 = True

check()

def is\_int\_saucejs\_2(event):

global parbaude\_saucejs\_2

parbaude = input\_saucejs\_2.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

parbaude = 1 / parbaude

except:

if input\_saucejs\_2.get() == "":

input\_saucejs\_2.config(bg="white")

submit\_button['state'] = tkinter.DISABLED

parbaude\_saucejs\_2 = False

else:

input\_saucejs\_2.config(bg="red")

submit\_button['state'] = tkinter.DISABLED

parbaude\_saucejs\_2 = False

else:

input\_saucejs\_2.config(bg="white")

parbaude\_saucejs\_2 = True

check()

def check(): # Pārbaude

if parbaude\_skaititajs\_1 and parbaude\_skaititajs\_2 and parbaude\_saucejs\_1 and parbaude\_saucejs\_2: # Ja visos lodziņos ir ierakstīti reāli skaitļi

submit\_button['state'] = tkinter.ACTIVE # tad poga kļūst aktīva

else:

submit\_button['state'] = tkinter.DISABLED # Ja kaut viena lodziņa nav ierakstīts reāls skaitļi, tad poga kļūst aktīva

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

logs = tkinter.Tk() # Tkinter (lai izmantotu to komandas)

logs.geometry("230x185") # Loga izmēra definalsēšana

logs.title("Darbības ar daļskaitļiem.") # Windows "loga" nosaukums

zimes\_veidi = ("+", "-", "x", ":")

# Labels

label\_dalisanas\_zime\_1 = tkinter.Label(logs, text="―", font=("Arial", 25)) #

label\_dalisanas\_zime\_1.place(x=19, y=50)

label\_dalisanas\_zime\_2 = tkinter.Label(logs, text="―", font=("Arial", 25)) #

label\_dalisanas\_zime\_2.place(x=71, y=50)

label\_dalisanas\_zime\_3 = tkinter.Label(logs, text="―", font=("Arial", 25)) #

label\_dalisanas\_zime\_3.place(x=159, y=50)

label\_skaititajs\_result = tkinter.Label(logs, text="", width=3)

label\_skaititajs\_result.place(x=166, y=52)

label\_saucejs\_result = tkinter.Label(logs, text="", width=3)

label\_saucejs\_result.place(x=166, y=80)

label\_minus = tkinter.Label(logs, text="", width=0, font=("Arial", 25))

label\_minus.place(x=136, y=50)

input\_skaititajs\_1 = tkinter.Entry(logs, width=3)

input\_skaititajs\_1.bind("<KeyRelease>", is\_int\_skaititajs\_1)

input\_skaititajs\_1.place(x=28, y=52)

input\_skaititajs\_2 = tkinter.Entry(logs, width=3)

input\_skaititajs\_2.bind("<KeyRelease>", is\_int\_skaititajs\_2)

input\_skaititajs\_2.place(x=80, y=52)

input\_saucejs\_1 = tkinter.Entry(logs, width=3)

input\_saucejs\_1.bind("<KeyRelease>", is\_int\_saucejs\_1)

input\_saucejs\_1.place(x=28, y=80)

input\_saucejs\_2 = tkinter.Entry(logs, width=3)

input\_saucejs\_2.bind("<KeyRelease>", is\_int\_saucejs\_2)

input\_saucejs\_2.place(x=80, y=80)

submit\_button = tkinter.Button(logs, text="=", width=20, bd=1, command=result)

submit\_button.place(x=103, y=65, width=25) # Parādam, kur poga tiks attēlota

submit\_button['state'] = tkinter.DISABLED # pēc noklusējuma poga ir izslēgta

button = tkinter.Button(logs, text="+", width=20, bd=1, command=mainit\_zimi)

button.place(x=53, y=65, width=25) # Parādam, kur poga tiks attēlota

parbaude\_skaititajs\_1 = False

parbaude\_skaititajs\_2 = False

parbaude\_saucejs\_1 = False

parbaude\_saucejs\_2 = False

logs.mainloop() # Obligāta rindiņa, lai logs butu redzāms visu laiku

**Testa piemēri:**

1)

A picture containing chart

Description automatically generated

2)

Chart, waterfall chart

Description automatically generated

3)

Chart, waterfall chart

Description automatically generated

4)

A picture containing graphical user interface

Description automatically generated

5)

A picture containing diagram

Description automatically generated

6)

A picture containing diagram

Description automatically generated

7)

A picture containing diagram

Description automatically generated

8)

A picture containing chart

Description automatically generated

9)

A picture containing graphical user interface

Description automatically generated

10)

Chart, waterfall chart

Description automatically generated

11)

A picture containing chart

Description automatically generated

12)

A picture containing diagram

Description automatically generated

**4. uzdevums**

Sastādīt programmu, kas organizē viendimensijas masīva izveidi, datu ievadi un izvadi, lielākās un mazākās vērtības atrašanu. Masīva izmēru N ievada lietotājs.

**Kods:**

# Programmas nosaukums: Viendimensijas masīva izveidi.

# 4. uzdevums (1MPR04\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas organizē viendimensijas masīva izveidi, datu ievadi un izvadi, lielākās un mazākās vērtības atrašanu. Masīva izmēru N ievada lietotājs.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

n = input("Ievadiet masīva izmēru N ===> ")

while is\_natural(n) == False:

n = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

n = int(n)

a = numpy.arange(n)

for i in range(n):

a[i] = int(input("Ievadi " + str(i) + ".elementu ===> "))

print("Masīva elementi ievadīšanas secībā:")

for i in range(n):

print(a[i])

print("Masīva minimālais elements: " + str(min(a)))

print("Masīva maksimālais elements: " + str(max(a)))

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

Text

Description automatically generated

4)

A screenshot of a computer

Description automatically generated with low confidence

5)

Text

Description automatically generated

**5. uzdevums**

Sastādīt programmu, kas demonstrē masīvu uzglabāšanu datora atmiņā, t.i., darbības ar diviem un vairākiem masīviem – viena masīva atsevišķu elementu piešķiršanu citiem masīva elementiem un viena masīva visu elementu piešķiršanu citam masīvam (ar tikpat, mazāku un lielāku elementu skaitu). Masīva elementa indeksa saturiskā jēga, norāde uz masīvu.

**Uzdevums tika izpildīts 01.03.2023. praktisko nodarbības laikā. Tie kas ir bijusi uz nodarbības tika atbrīvoti no šī uzdevuma pildīšanas.**

**PU1.**

Darbības ar “jauktiem daļskaitļiem” pēc 2 uzdevuma nosacījumiem. Rezultātā jābūt redzamam “pareizam” skaitlim.

**Kods:**

# Programmas nosaukums: Četras aritmētiskās darbības ar daļskaitļiem.

# Papilduzdevums 1 (1MPR04\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas realizē četras aritmētiskās darbības ar daļskaitļiem. Piespiežot pogu =, tiek parādīts rezultāts atbilstoši, tai darbības zīmei, kas redzama uz pogas.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

import tkinter # Importējam tkinter moduli

# ----------- Pirmās daļas pārbaudīšana --------------

def is\_skaititajs\_1\_mazaks\_saucejs\_1(a1, d1):

global parbaude\_skaititajam\_1

if a1 <= d1:

parbaude\_skaititajam\_1 = True

else:

parbaude\_skaititajam\_1 = False

def is\_skaititajs\_2\_mazaks\_saucejs\_2(b1, e1):

global parbaude\_skaititajam\_2

if b1 <= e1:

parbaude\_skaititajam\_2 = True

else:

parbaude\_skaititajam\_2 = False

def is\_pareizs\_vesela\_dala\_1():

global parbaude\_veselais\_1

parbaude = input\_veselais\_1.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

except:

parbaude\_veselais\_1 = False

else:

parbaude\_veselais\_1 = True

def is\_pareizs\_skaititajs\_1():

global parbaude\_skaititajam\_1, parbaude\_saucejam\_1

parbaude = input\_skaititajs\_1.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

except:

parbaude\_skaititajam\_1 = False

else:

a1 = input\_skaititajs\_1.get()

d1 = input\_skaititajs\_2.get()

is\_skaititajs\_1\_mazaks\_saucejs\_1(a1, d1)

def is\_pareizs\_saucejs\_1():

global parbaude\_skaititajam\_1, parbaude\_saucejam\_1

parbaude = input\_skaititajs\_2.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

parbaude = 1 / parbaude

except:

parbaude\_saucejam\_1 = False

else:

d1 = input\_skaititajs\_2.get()

a1 = input\_skaititajs\_1.get()

is\_skaititajs\_1\_mazaks\_saucejs\_1(a1, d1)

parbaude\_saucejam\_1 = True

# ---------------------------------------------------

# ----------- Otrās daļas pārbaudīšāna --------------

def is\_pareiza\_vesela\_dala\_2():

global parbaude\_veselais\_2

parbaude = input\_veselais\_2.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

except:

parbaude\_veselais\_2 = False

else:

parbaude\_veselais\_2 = True

def is\_pareizs\_skaititajs\_2():

global parbaude\_skaititajam\_2

parbaude = input\_saucejs\_1.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

except:

parbaude\_skaititajam\_2 = False

else:

b1 = input\_saucejs\_1.get()

e1 = input\_saucejs\_2.get()

is\_skaititajs\_2\_mazaks\_saucejs\_2(b1, e1)

def is\_pareizs\_saucejs\_2():

global parbaude\_skaititajam\_2, parbaude\_saucejam\_2

parbaude = input\_saucejs\_2.get()

try:

parbaude = int(parbaude)

parbaude = math.sqrt(parbaude)

parbaude = 1 / parbaude

except:

parbaude\_saucejam\_2 = False

else:

if input\_saucejs\_2.get() < input\_saucejs\_1.get():

parbaude\_skaititajam\_2 = False

else:

parbaude\_skaititajam\_2 = True

parbaude\_saucejam\_2 = True

# -------------------------

def check():

if parbaude\_skaititajam\_1 and parbaude\_skaititajam\_2 and parbaude\_saucejam\_1 and parbaude\_saucejam\_2 and parbaude\_veselais\_1 and parbaude\_veselais\_2: # Ja visos lodziņos ir ierakstīti reāli skaitļi

return True

else:

return False

# ----------------------------------------------------------------------------------------

# ----------------------------------------------------------------------------------------

# --------- Funkcijas reķināšanai ---------

def mainit\_zimi():

cetras\_zimes\_mainisana\_plus\_minus\_reizinat\_dalit(button)

def cetras\_zimes\_mainisana\_plus\_minus\_reizinat\_dalit(ee):

index = zimes\_veidi.index(ee['text'])

ee.config(text=zimes\_veidi[(index + 1) % 4])

def saskaitit(d1, d2):

saucejs = d1[1] \* d2[1]

skaititajs = d1[0] \* d2[1] + d1[1] \* d2[0]

return (skaititajs, saucejs)

def atnemt(d1, d2):

saucejs = d1[1] \* d2[1]

skaititajs = d1[0] \* d2[1] - d1[1] \* d2[0]

return (skaititajs, saucejs)

def reizinat(d1, d2):

skaititajs = d1[0] \* d2[0]

saucejs = d1[1] \* d2[1]

return (skaititajs, saucejs)

def dalit(d1, d2):

skaititajs = d1[0] \* d2[1]

saucejs = d1[1] \* d2[0]

return (skaititajs, saucejs)

def gcd(a, b):

while b != 0:

c = a % b

a = b

b = c

return a

def result():

is\_pareizs\_vesela\_dala\_1()

is\_pareizs\_skaititajs\_1()

is\_pareizs\_saucejs\_1()

is\_pareiza\_vesela\_dala\_2()

is\_pareizs\_skaititajs\_2()

is\_pareizs\_saucejs\_2()

if check():

result2()

else:

label\_error.config(text="Nepareizi ievadīti dati kāda no lodziņiem!")

label\_vesels\_skaitlis\_result.config(text="")

label\_skaititajs\_result.config(text="")

label\_saucejs\_result.config(text="")

label\_dalisanas\_zime\_3.config(text="")

parbaude\_veselais\_1 = False

parbaude\_skaititajam\_1 = False

parbaude\_saucejam\_1 = False

parbaude\_veselais\_2 = False

parbaude\_skaititajam\_2 = False

parbaude\_saucejam\_2 = False

def result2():

label\_error.config(text="")

minus = ""

index = zimes\_veidi.index(button['text'])

label\_skaititajs\_result.config(text="")

label\_saucejs\_result.config(text="")

label\_minus.config(text="")

d1 = [int(input\_veselais\_1.get()), int(input\_skaititajs\_1.get()), int(input\_skaititajs\_2.get())] # 1 - vesela daļa, 2 - skaititajs, 3 - saucejs

d2 = [int(input\_veselais\_2.get()), int(input\_saucejs\_1.get()), int(input\_saucejs\_2.get())] # 1 - vesela daļa, 2 - skaititajs, 3 - saucejs

d1[1] = d1[0] \* d1[2] + d1[1]

d2[1] = d2[0] \* d2[2] + d2[1]

neista\_dala1 = (d1[1], d1[2]) # konvērtē uz neīstu daļu 1.daļu. Nav veselas daļas

neista\_dala2 = (d2[1], d2[2]) # konvērtē uz neīstu daļu 2.daļu. Nav veselas daļas

if index == 0:

finals = saskaitit(neista\_dala1, neista\_dala2)

elif index == 1:

finals = atnemt(neista\_dala1, neista\_dala2)

if finals[0] < 0:

finals = (abs(finals[0]), finals[1])

minus = "-"

elif index == 2:

neista\_dala1\_list = list(neista\_dala1)

neista\_dala1\_list[0] = d1[0] \* neista\_dala1[1] + neista\_dala1[0]

neista\_dala1 = tuple(neista\_dala1)

finals = reizinat(neista\_dala1, neista\_dala2)

else:

finals = dalit(neista\_dala1, neista\_dala2)

gcd1 = gcd(finals[0], finals[1]) # Daļas saīsināšānai

if gcd1 == 0:

label\_vesels\_skaitlis\_result.config(text="undef") # Dalīšana ar 0

label\_skaititajs\_result.config(text="")

label\_saucejs\_result.config(text="")

label\_dalisanas\_zime\_3.config(text="")

else:

parbaude\_skaititajam\_1 = finals[0] // gcd1

parbaude\_skaititajam\_2 = finals[1] // gcd1

list\_dala\_beigas = [0, parbaude\_skaititajam\_1, parbaude\_skaititajam\_2]

if list\_dala\_beigas[2] == 0:

label\_vesels\_skaitlis\_result.config(text="undef")

label\_skaititajs\_result.config(text="")

label\_saucejs\_result.config(text="")

label\_dalisanas\_zime\_3.config(text="")

else:

vesela\_dala = list\_dala\_beigas[1] // list\_dala\_beigas[2]

skaititajs = list\_dala\_beigas[1] % list\_dala\_beigas[2]

saucejs = list\_dala\_beigas[2]

list\_dala\_beigas = (vesela\_dala, skaititajs, saucejs)

label\_vesels\_skaitlis\_result.config(text=str(vesela\_dala))

label\_skaititajs\_result.config(text=str(skaititajs))

label\_saucejs\_result.config(text=str(saucejs))

label\_minus.config(text=minus)

label\_dalisanas\_zime\_3.config(text="―", font=("Arial", 25))

# ----- Testēšanai -----

#print("Finālas daļas veselā daļa: " + str(list\_dala\_beigas[0]))

#print("Finālas daļas skaitītājs: " + str(list\_dala\_beigas[1]))

#print("Finālas daļas saucējs: " + str(list\_dala\_beigas[2]))

# ----- Testēšanai -----

if list\_dala\_beigas[1] == 0 and list\_dala\_beigas[2] != 0: # Ja ir 0 skaitītāja un nav 0 saucēja, tad nerakstam daļsvītru un nodzēsam vērtības

label\_skaititajs\_result.config(text="")

label\_saucejs\_result.config(text="")

label\_dalisanas\_zime\_3.config(text="")

elif vesela\_dala == 0: # Ja vesela daļā ir 0, tad nav jēgas to rakstīt.

label\_vesels\_skaitlis\_result.config(text="")

parbaude\_veselais\_1 = False

parbaude\_skaititajam\_1 = False

parbaude\_saucejam\_1 = False

parbaude\_veselais\_2 = False

parbaude\_skaititajam\_2 = False

parbaude\_saucejam\_2 = False

for widget in logs.winfo\_children():

if isinstance(widget, tkinter.Entry): # Ja tas ir entry widgets

widget.delete(0, 'end') # tad nodzes visus entry

# --------- Funkcijas reķināšanai ---------

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

logs = tkinter.Tk() # Tkinter (lai izmantotu to komandas)

logs.geometry("230x185") # Loga izmēra definēšana

logs.title("Daļskaitļi") # Windows "loga" nosaukums

# Labels dalīšanas zīmes

label\_dalisanas\_zime\_1 = tkinter.Label(logs, text="―", font=("Arial", 25)) #

label\_dalisanas\_zime\_1.place(x=19, y=50)

label\_dalisanas\_zime\_2 = tkinter.Label(logs, text="―", font=("Arial", 25)) #

label\_dalisanas\_zime\_2.place(x=91, y=50)

label\_dalisanas\_zime\_3 = tkinter.Label(logs, text="", font=("Arial", 25)) #

label\_dalisanas\_zime\_3.place(x=159, y=50)

# Rezultāts

label\_vesels\_skaitlis\_result = tkinter.Label(logs, text="", width=3)

label\_vesels\_skaitlis\_result.place(x=156, y=65)

label\_skaititajs\_result = tkinter.Label(logs, text="", width=3)

label\_skaititajs\_result.place(x=176, y=52)

label\_saucejs\_result = tkinter.Label(logs, text="", width=3)

label\_saucejs\_result.place(x=176, y=80)

label\_error = tkinter.Label(logs, text="", width=0)

label\_error.place(x=5, y=10)

# Munis priekšā

label\_minus = tkinter.Label(logs, text="", width=0, font=("Arial", 25))

label\_minus.place(x=150, y=50)

# Entry logi

input\_veselais\_1 = tkinter.Entry(logs, width=3)

input\_veselais\_1.place(x=2, y=65)

input\_skaititajs\_1 = tkinter.Entry(logs, width=3)

input\_skaititajs\_1.place(x=28, y=52)

input\_saucejs\_1 = tkinter.Entry(logs, width=3)

input\_saucejs\_1.place(x=100, y=52)

input\_veselais\_2 = tkinter.Entry(logs, width=3)

input\_veselais\_2.place(x=77, y=65)

input\_skaititajs\_2 = tkinter.Entry(logs, width=3)

input\_skaititajs\_2.place(x=28, y=80)

input\_saucejs\_2 = tkinter.Entry(logs, width=3)

input\_saucejs\_2.place(x=100, y=80)

# -----------------------------------------

# Rezultāta poga "="

submit\_button = tkinter.Button(logs, text="=", width=20, bd=1, command=result) # Izmantojam definētas komandas, lai pēc pogas nospiešanas tā komanda tiek izpildīta

submit\_button.place(x=123, y=65, width=25) # Parādam, kur poga tiks attēlota

# Mainīt zīmi poga (+, -, x, :)

zimes\_veidi = ("+", "-", "x", ":")

button = tkinter.Button(logs, text="+", width=20, bd=1, command=mainit\_zimi) # Izmantojam definētas komandas, lai pēc pogas nospiešanas tā komanda tiek izpildīta

button.place(x=53, y=65, width=25) # Parādam, kur poga tiks attēlota

parbaude\_veselais\_1 = False

parbaude\_skaititajam\_1 = False

parbaude\_saucejam\_1 = False

parbaude\_veselais\_2 = False

parbaude\_skaititajam\_2 = False

parbaude\_saucejam\_2 = False

logs.mainloop()

**Testa piemēri:**

1)

A picture containing chart

Description automatically generated

A picture containing chart

Description automatically generated

2)

Chart, waterfall chart

Description automatically generated

Chart

Description automatically generated

3)

Chart

Description automatically generated with low confidence

Chart

Description automatically generated with medium confidence

4)

A picture containing graphical user interface

Description automatically generated

Chart, waterfall chart

Description automatically generated

5)

A picture containing chart

Description automatically generated

Chart, waterfall chart

Description automatically generated

6)

A picture containing graphical user interface

Description automatically generated

Chart, waterfall chart

Description automatically generated with medium confidence

7)

A picture containing graphical user interface

Description automatically generated

Chart, waterfall chart

Description automatically generated

8)

A picture containing graphical user interface

Description automatically generated

Chart, waterfall chart

Description automatically generated

9)

A picture containing graphical user interface

Description automatically generated

Chart

Description automatically generated

10)

A picture containing graphical user interface

Description automatically generated

Chart, waterfall chart

Description automatically generated

11)

A picture containing chart

Description automatically generated

Chart, waterfall chart

Description automatically generated

12)

A picture containing chart

Description automatically generated

Chart, waterfall chart

Description automatically generated