**1MPR13\_Simona\_Bļinova sb24037**

**1.uzdevums**

Programma, kas realizē veikala simulāciju.

**Kods:**

class Prece:

def \_\_init\_\_(self, artikuls, nosaukums, daudzums, cena):

self.artikuls = artikuls

self.nosaukums = nosaukums

self.daudzums = daudzums

self.cena = cena

def \_\_str\_\_(self):

return f'{self.nosaukums} (Artikuls: {self.artikuls}) - {self.daudzums} gab., {self.cena:.2f} EUR'

class Veikals:

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

self.krajumi = {} # key: preces nosaukums, vertiba: prece objekts

def pievienot\_preci(self, prece):

if prece.nosaukums in self.krajumi:

self.krajumi[prece.nosaukums].daudzums += prece.daudzums

else:

self.krajumi[prece.nosaukums] = prece

def paradit\_preces(self):

print('Pieejamas preces veikalā:')

for prece in self.krajumi:

print(prece)

def ir\_prece(self, nosaukums, daudzums):

return nosaukums in self.krajumi and self.krajumi[nosaukums].daudzums >= daudzums

def pardod\_preci(self, nosaukums, daudzums):

if self.ir\_prece(nosaukums, daudzums):

prece = self.krajumi[nosaukums]

prece.daudzums -= daudzums

return Prece(prece.artikuls, nosaukums, daudzums, prece.cena)

else:

print(f'Nav pietiekami daudz preces: {nosaukums}')

return None

class Grozs:

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

self.pirkumi = []

def pievienot\_pirkumu(self, prece):

self.pirkumi.append(prece)

def izdrukat\_cekus(self):

print('Pirkuma čeks:')

kopa = 0

for prece in self.pirkumi:

summa = prece.daudzums \* prece.cena

print(f'{prece.nosaukums} x {prece.daudzums} = {summa:.2f} EUR')

kopa += summa

print(f'Kopa jamaksa: {kopa:.2f} EUR')

def galvena\_programma():

veikals = Veikals()

print('~~~ Preču piegāde veikalām ~~~')

while True:

artikuls = input('\nIevadiet artikulu --> ')

nosaukums = input('Ievadiet preces nosaukumu --> ')

daudzums = int(input('Ievadiet daudzumu --> '))

cena = float(input('Ievadiet preces cenu (EUR) --> '))

veikals.pievienot\_preci(Prece(artikuls, nosaukums, daudzums, cena))

paz = input('\nVai pievienot vēl preces? (j/n)')

if paz == 'n':

break

print(' ')

veikals.paradit\_preces()

print('\n~~~ Iepirkšanas ~~~')

grozs = Grozs()

while True:

nosaukums = input('\nIevadiet preces nosaukumu --> ')

daudzums = int(input('Ievadiet daudzumu --> '))

if veikals.ir\_prece(nosaukums, daudzums):

prece = veikals.pardod\_preci(nosaukums, daudzums)

grozs.pievienot\_pirkumu(prece)

else:

print('Nav pieejams!')

paz = input('\n Vai pievienot vēl preces? (j/n)')

if paz == 'n':

break

print('\n~~~ Grozs ~~~')

print(' ')

grozs.izdrukat\_cekus()

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

galvena\_programma()

Testa piemērs(1)

A screenshot of a computer program

AI-generated content may be incorrect.

Testa piemērs(2)

A screenshot of a computer program

AI-generated content may be incorrect.

Testa piemērs(3)

A screenshot of a computer program

AI-generated content may be incorrect.

**2.uzdevums**

Programma, kas realizē darbības ar kompleksiem skaitļiem.

**Kods:**

import math

class Kompleksais\_skaitlis:

def \_\_init\_\_(self, a, b):

self.z = (a, b)

def \_\_str\_\_(self):

if self.z[0] == 0 and self.z[1] == 0:

return '0'

elif self.z[0] == 0:

return f'{self.z[1]}i'

elif self.z[1] == 0:

return f'{self.z[0]}'

elif self.z[1] < 0:

return f'{self.z[0]} - {abs(self.z[1])}i'

else:

return f'{self.z[0]} + {self.z[1]}i'

class Darbibas:

def \_\_init\_\_(self, sk1, sk2):

self.darb = {}

self.darb['sk1'] = sk1

self.darb['sk2'] = sk2

def saistitais(self):

self.darb['saist1'] = Kompleksais\_skaitlis(self.darb['sk1'].z[0], -self.darb['sk1'].z[1])

self.darb['saist2'] = Kompleksais\_skaitlis(self.darb['sk2'].z[0], -self.darb['sk2'].z[1])

def modulis(self):

self.darb['modulis1'] = str(math.sqrt(self.darb['sk1'].z[0]\*\*2 + self.darb['sk1'].z[1]\*\*2))

self.darb['modulis2'] = str(math.sqrt(self.darb['sk2'].z[0]\*\*2 + self.darb['sk2'].z[1]\*\*2))

def summa(self):

a = self.darb['sk1'].z[0] + self.darb['sk2'].z[0]

b = self.darb['sk1'].z[1] + self.darb['sk2'].z[1]

self.darb['summa'] = Kompleksais\_skaitlis(a, b)

def reizinajums(self):

a = self.darb['sk1'].z[0] \* self.darb['sk2'].z[0] - self.darb['sk1'].z[1] \* self.darb['sk2'].z[1]

b = self.darb['sk1'].z[0] \* self.darb['sk2'].z[1] + self.darb['sk2'].z[0] \* self.darb['sk1'].z[1]

self.darb['reizinajums'] = Kompleksais\_skaitlis(a, b)

def dalijums(self):

if self.darb['sk2'].z[0] != 0 and self.darb['sk2'].z[1] != 0:

saucejs = self.darb['sk2'].z[0] \*\* 2 + self.darb['sk2'].z[1] \*\* 2

a\_skait = self.darb['sk1'].z[0] \* self.darb['sk2'].z[0] + self.darb['sk1'].z[1] \* self.darb['sk2'].z[1]

b\_skait = self.darb['sk2'].z[0] \* self.darb['sk1'].z[1] - self.darb['sk1'].z[0] \* self.darb['sk2'].z[1]

self.darb['dalijums'] = Kompleksais\_skaitlis(a\_skait/saucejs, b\_skait/saucejs)

else:

self.darb['dalijums'] = '---'

def galvena\_programma():

a1 = float(input('Ievadiet reālo koeficientu 1. skaitlim --> '))

b1 = float(input('Ievadiet imaginaro koeficientu 1. skaitlim --> '))

skaitlis1 = Kompleksais\_skaitlis(a1, b1)

a2 = float(input('Ievadiet reālo koeficientu 2. skaitlim --> '))

b2 = float(input('Ievadiet imaginaro koeficientu 2. skaitlim --> '))

skaitlis2 = Kompleksais\_skaitlis(a2, b2)

print('')

darbibas = Darbibas(skaitlis1, skaitlis2)

darbibas.saistitais()

darbibas.modulis()

darbibas.summa()

darbibas.reizinajums()

darbibas.dalijums()

vertibas = []

for vertiba in darbibas.darb.values():

vertibas.append(vertiba)

print('Pirmais skaitlis:', vertibas[0])

print('Otrais skaitlis:', vertibas[1])

print('Pirmā skaitļa saistītais:', vertibas[2])

print('Otrā skaitļa saistītais:', vertibas[3])

print('Pirmā skaitļa modulis:', vertibas[4])

print('Otrā skaitļa modulis:', vertibas[5])

print('Skaitļu summa:', vertibas[6])

print('Skaitļu reizinājums:', vertibas[7])

print('Skaitļu dalījums:', vertibas[8])

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

galvena\_programma()

Testa piemērs(1)

A computer screen shot of white text

AI-generated content may be incorrect.

Testa piemērs(2)

A computer screen shot of white text

AI-generated content may be incorrect.

Testa piemērs(3)

A computer screen shot of white text

AI-generated content may be incorrect.

**3.uzdevums**

Programma, kas aprēķina n-stūra laukumu.

**Kods:**

class Virsotne:

def \_\_init\_\_(self, x, y):

self.koordinatas = (x, y)

class Nsturis:

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

self.virsotnes = virsotnes

def pievienot\_virsotni(self, x, y):

self.virsotnes.append(Virsotne(x, y))

def laukums(self):

if len(self.virsotnes) < 3:

return 'Virsotņu skaits nav pietiekams laukuma aprēķinam.'

else:

summa = 0

for i in range(len(self.virsotnes)-1):

#print(self.virsotnes[i].koordinatas[0], self.virsotnes[i+1].koordinatas[0])

s = (self.virsotnes[i].koordinatas[0] + self.virsotnes[i+1].koordinatas[0]) \* (self.virsotnes[i].koordinatas[1] - self.virsotnes[i+1].koordinatas[1])

summa += s

abs\_summa = abs(summa)

return abs\_summa / 2

def galvena\_programma():

print('N-stūra virsotņu ievade')

print('')

v = []

nsturis = Nsturis(v)

while True:

koord\_x = float(input('Ievadiet x koordinātu --> '))

koord\_y = float(input('Ievadiet y koordinātu --> '))

nsturis.pievienot\_virsotni(koord\_x, koord\_y)

print('')

turp = input('Vēlaties turpināt ievadi? (t - turpināt, p - pabeigt) --> ')

print('')

if turp == 'p':

break

print('')

print('Laukuma aprēķins')

print('')

print(f'Laukums: {nsturis.laukums()}')

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

galvena\_programma()

Testa piemērs(1)

A screenshot of a computer program

AI-generated content may be incorrect.

Testa piemērs(2)

A screen shot of a computer program

AI-generated content may be incorrect.

Testa piemērs(3)

A screen shot of a computer program

AI-generated content may be incorrect.