1MPR12_Simona_Bļinova sb24037

1.uzdevums

Programma, kas realizē Nomas punkta darbību.

Kods:

```
auto_id = set()
nomatie_id = set()

############## NOMAS PUNKTS ###############
class Nomas_punkts:
  '''
  Klase 'Nomas_punkts' pārstāv vienkāršu nomas punkta modeli ar pamatinformāciju:
    - nomas punkta telefona numurs
    - nomas punkta email
    - visi auto
    - iznomatie auto ar klientu informāciju

  Klase nodrošina metodes:
    - jauna auto iepirkšanai
    - auto utilizēšanai
    - auto atdošanai nomai
    - auto saņemšanu atpakaļ
    - visu auto izvadei
    - iznomato auto izvadei ar klienta informāciju
    - nomas punkta kontaktu izvade

  Lauki:
    rekveziti (str): Nomas punkta telefona numurs
    auto (list): Saraksts ar visiem pieejamiem auto
    iznomatie_auto (list): Saraksts ar kortežiem (tuple), kuros viens elements ir iznomāta auto id un klienta informāciju

  Metodes:
    pirkt()
    utilizet()
    iznomat()
    sanemt()
    visi_auto()
    iznomatie()
    kontakti()
```

```
  '''

  def __init__(self, auto, iznomatie_auto):
    '''
    Konstruktors.
    Piešķir sākotnējas vērtības laukiem.

    Argumenti:
      auto (list): Saraksts ar visiem pieejamiem auto
      iznomatie_auto (list): Saraksts ar kortežiem (tuple), kuros viens elements ir iznomāta auto id
un klienta informāciju
    '''

    self.tel = '+37100000000'
    self.email = 'nomaspunkts@inbox.lv'
    self.auto = auto
    self.iznomatie_auto = iznomatie_auto

  def pirkt(self):
    '''
    Metode.
    Pievieno jaunu auto nomas punktam.
    '''

    # id aprēķins
    l = len(self.auto)

    for i in range(1, l+1):
      if i not in auto_id:
        id = i
        break
    else:
      id = l+1
    auto_id.add(id)

    m = input('Ievadiet auto marku --> ')
    t = input('Ievadiet auto tipu (k - krāvas, v - vieglais, a - autobuss) --> ')

    match t:
      case 'k':
        sv = int(input('Ievadiet krāvas auto maksimālo pārvadājamo svaru tonnās --> '))
```

```
            self.auto.append(Kravas(id, m, 'krāvas', sv))
        case 'v':
            k = input('Ievadiet ātrumkārbas tipu --> ')
            ps = int(input('Ievadiet maksimālo pārvadājamo pasažieru skaitu --> '))
            self.auto.append(Vieglais(id, m, 'vieglais', k, ps))
        case 'a':
            ps = int(input('Ievaidet maksimālo pārvadājamo pasažieru skaitu --> '))
            self.auto.append(Autobuss(id, m, 'autobuss', ps))
        case _:
            print('Tāda tipa mašīnu nevar nopirkt!')

def utilizet(self):
    l = len(self.auto)
    #print(l)

    a = int(input('Ievadiet utilizejamas mašīnas id --> '))

    if l == 0:
        print('Nomā nav auto!')
    else:
        for i in range(l):
            if self.auto[i].id == a:
                self.auto.pop(i)
                auto_id.remove(a)
                break

def iznomat(self):
    '''
    Metode.
    Atdod auto nomai.
    '''
    l = len(self.auto)

    i_d = int(input('Ievadiet iznomātas mašīnas id --> '))

    if l == 0:
        print('Nomā nav auto!')
    else:
        v = input('Ievadiet vārdu --> ')
        u = input('Ievadiet uzvārdu --> ')
        tel = input('Ievadiet telefona numuru --> ')
```

```
    t = input('Ievaidet personas tipu (f - fiziska, j - juridiska) --> ')

    for i in range(l):
      if self.auto[i].id ==i_d:
        a = self.auto[i]
        break

    match t:
      case 'f':
        k = input('Ievadiet vadītāja kategoriju (B - vieglie, C - krāvas, D - autobuss) --> ')
        kl = Fiziska(v, u, 'fiziska persona', tel, k)

        if a.tips == 'autobuss' and k == 'D':
          nomatie_id.add(a.id)
          self.auto.pop(i)
          self.iznomatie_auto.append((a, kl))
        elif a.tips == 'krāvas' and k == 'C':
          nomatie_id.add(a.id)
          self.auto.pop(i)
          self.iznomatie_auto.append((a, kl))
        elif a.tips == 'vieglais' and k == 'B':
          nomatie_id.add(a.id)
          self.auto.pop(i)
          self.iznomatie_auto.append((a, kl))
        else:
          print('Kategorija nav pietiekama!')



      case 'j':
        o = input('Ievadiet pārstāvētu organizāciju --> ')
        kl = Juridiska(v, u, 'juridiska persona', tel, o)
        nomatie_id.add(a.id)
        self.auto.pop(i)
        self.iznomatie_auto.append((a, kl))

def sanemt(self):
  '''
  Metode.
  Auto saņemšanai atpakaļ.
  '''
  l = len(self.iznomatie_auto)
```

```
    if l == 0:
        print('Nav iznomāto auto!')
    else:
        i_d = int(input('Ievadiet saņēmamas auto id --> '))

        if i_d not in nomatie_id:
            print('Šīs auto nav atdots nomai!')
        else:
            for i in range(l):
                if self.iznomatie_auto[i][0].id == i_d:
                    nomatie_id.remove(i_d)
                    self.auto.append(self.iznomatie_auto[i][0])
                    self.iznomatie_auto.pop(i)
                    break

def visi_auto(self):
    '''
    Metode.
    Izraksta visas auto.
    '''
    l = len(self.auto)

    if l == 0:
        print('Nomā nav auto!')
    else:
        for i in range(l):
            print(self.auto[i])

def iznomatie(self):
    '''
    Metode.
    Izvada visus iznomatus auto un informāciju par klientu.
    '''
    l = len(self.iznomatie_auto)

    if l == 0:
        print('Nav iznomāto auto!')
    else:
        for i in range(l):
            print(self.iznomatie_auto[i][1])
```

```
            print(self.iznomatie_auto[i][0])
            if i != l-1:
                print(' ')

    def kontakti(self):
        '''
        Metode.
        Izvada nomas punkta kontaktus.
        '''
        print(f'Nomas punkta tel.: {self.tel}, email: {self.email}')

############## AUTO ######################
class Auto:
    '''
    Klase 'Auto' pārstāv vienkāršu auto modeli ar pamatinformāciju:
        - auto id
        - auto marka
        - auto tips (krāvas, vieglais, autobuss)

    Lauki:
        id (int): Auto individuāls identifikācijas numurs
        marka (str): Auto marka
        tips (str): Auto tips
    '''

    def __init__(self, id, marka, tips):
        '''
        Konstruktors.
        Piešķir sākotnējas vērtības laukiem.

        Argumenti:
            id (int): Auto individuāls identifikācijas numurs
            marka (str): Saraksts ar visiem pieejamiem auto
            tips (str): Saraksts ar kortežiem (tuple), kuros viens elements ir iznomāta auto id un klienta
informāciju
        '''

        self.id = id
        self.marka = marka
        self.tips = tips
```

```
class Kravas(Auto):
    '''
    Klase 'Kravas' pārstāv krāvas auto modeli ar pamatinformāciju:
        Informācija no vecāka klases Auto:
            - auto id
            - auto marka
            - auto tips (krāvas, vieglais, autobuss)
        - maksimālais pārvādamais svars tonnās

    Klase nodrošina metodes:
        - visas informācijas izvadei

    Lauki:
        No vecāka klases Auto:
            id (int): Auto individuāls identifikācijas numurs
            marka (str): Auto marka
            tips (str): Auto tips
        svars (int): Krāvas auto maksimālais pārvadāmais svars tonnās

    Metodes:
        __str__()
    '''

    def __init__(self, id, marka, tips, svars):
        '''
        Konstruktors.
        Piešķir sākotnējas vērtības laukiem.

        Argumenti:
            No vecāka klases Auto:
                id (int): Auto individuāls identifikācijas numurs
                marka (str): Auto marka
                tips (str): Auto tips
            svars (int): Krāvas auto maksimālais pārvadāmais svars tonnās
        '''

        super().__init__(id, marka, tips)
        self.svars = svars

    def __str__(self):
```

```
        '''
        Metode.
        Izraksta visu informāciju par krāvas auto.
        '''

        return f'Krāvas auto {self.marka} nr.{self.id}: {self.marka}, max parvad. svars - {self.svars} tonnas.'


class Vieglais(Auto):
    '''
    Klase 'Vieglais' pārstāv viegla auto modeli ar pamatinformāciju:
        Informācija no vecāka klases Auto:
            - auto id
            - auto marka
            - auto tips (krāvas, vieglais, autobuss)
        - ātrumkārbas veidu
        - max pārvaidājamo pasažieru skaits

    Klase nodrošina metodes:
        - visas informācijas izvadei

    Lauki:
        No vecāka klases Auto:
            id (int): Auto individuāls identifikācijas numurs
            marka (str): Auto marka
            tips (str): Auto tips
        karba (str): Ātrumkārbas veids
        pasazieru_skaits (int): Viegla auto maksimālais pārvadāmais pasažieru skaits

    Metodes:
        __str__()
    '''

    def __init__(self, id, marka, tips, karba, pasazieru_skaits):
        '''
        Konstruktors.
        Piešķir sākotnējas vērtības laukiem.

        Argumenti:
            No vecāka klases Auto:
```

```
            id (int): Auto individuāls identifikācijas numurs
            marka (str): Auto marka
            tips (str): Auto tips
        karba (int): Viegla auto maksimālais pārvadāmais svars tonnās
        pasazieru_skaits (int): Viegla auto maksimālais pārvadāmais pasažieru skaits
    '''

    super().__init__(id, marka, tips)
    self.karba = karba
    self.pasazieru_skaits = pasazieru_skaits

  def __str__(self):
    '''
    Metode.
    Izraksta visu informāciju par vieglo auto.
    '''

    return f'Vieglais auto {self.marka} nr.{self.id}, ātrumkārba {self.karba}, max pārvad. pasažieru
skaits {self.pasazieru_skaits}'


class Autobuss(Auto):
  '''
  Klase 'Autobuss' pārstāv autobusa modeli ar pamatinformāciju:
    Informācija no vecāka klases Auto:
      - auto id
      - auto marka
      - auto tips (krāvas, vieglais, autobuss)
    - max pārvaidājamo pasažieru skaits

  Klase nodrošina metodes:
    - visas informācijas izvadei

  Lauki:
    No vecāka klases Auto:
      id (int): Auto individuāls identifikācijas numurs
      marka (str): Auto marka
      tips (str): Auto tips
    pasazieru_skaits (int): Autobusa maksimālais pārvadāmais pasažieru skaits

  Metodes:
```

```
    __str__()
  '''

  def __init__(self, id, marka, tips, pasazieru_skaits):
    '''
    Konstruktors.
    Piešķir sākotnējas vērtības laukiem.

    Argumenti:
      No vecāka klases Auto:
        id (int): Auto individuāls identifikācijas numurs
        marka (str): Auto marka
        tips (str): Auto tips
      pasazieru_skaits (int): Autobusa maksimālais pārvadāmais pasažieru skaits
    '''

    super().__init__(id, marka, tips)
    self.pasazieru_skaits = pasazieru_skaits

  def __str__(self):
    '''
    Metode.
    Izraksta visu informāciju par autobusu.
    '''

    return f'Autobuss {self.marka} nr.{self.id}: max pārvad. pasažieru skaits {self.pasazieru_skaits}'


########### KLIENTI ######################
class Klienti:
  '''
  Klase 'Klienti' pārstāv vienkāršu klienta modeli ar pamatinformāciju:
    - vārds
    - uzvārds
    - tips (fiziska vai juridiska persona)

  Lauki:
    vards (str): Klienta vārds
    uzvards (str): Klienta uzvārds
    tips (str): Klienta tips
  '''
```

```
    def __init__(self, vards, uzvards, tips, telefons):
        '''
        Konstruktors.
        Piešķir sākotnējas vērtības laukiem.

        Argumenti:
            vards (str): Klienta vārds
            uzvards (str): Klienta uzvārds
            tips (str): Klienta tips
        '''

        self.vards = vards
        self.uzvards = uzvards
        self.tips = tips
        self.telefons = telefons


class Fiziska(Klienti):
    '''
    Klase 'Fiziska' pārstāv fiziskas personas modeli ar pamatinformāciju:
        Informācija no vecāka klases Auto:
            - vārds
            - uzvārds
            - tips (fiziska vai juridiska persona)
        - telefons
        - kategorija

    Klase nodrošina metodes:
        - visas informācijas izvadei

    Lauki:
        No vecāka klases Klienti:
            vards (str): Klienta vārds
            uzvards (str): Klienta uzvārds
            tips (str): Klienta tips
            telefons (str): Klienta kontakts
        kategorija (str): Fiziskas personas vadītāja kategorija

    Metodes:
        __str__()
```

```
    '''

    def __init__(self, vards, uzvards, tips, telefons, kategorija):
        '''
        Konstruktors.
        Piešķir sākotnējas vērtības laukiem.

        Argumenti:
            No vecāka klases Klienti:
                vards (str): Klienta vārds
                uzvards (str): Klienta uzvārds
                tips (str): Klienta tips
                telefons (str): Klienta kontakts
            kategorija (str): Fiziskas personas vadītāja kategorija
        '''

        super().__init__(vards, uzvards, tips, telefons)
        self.kategorija = kategorija

    def __str__(self):
        '''
        Metode.
        Izraksta visu informāciju par klientu.
        '''

        return f'Fiziska persona {self.vards} {self.uzvards}, tel.: {self.telefons}, kategorija:
{self.kategorija}'


class Juridiska(Klienti):
    '''
    Klase 'Juridiska' pārstāv juridiskas personas modeli ar pamatinformāciju:
        Informācija no vecāka klases Klienti:
            - vārds
            - uzvārds
            - tips (fiziska vai juridiska persona)
            - telefons
        - organizacija

    Klase nodrošina metodes:
        - visas informācijas izvadei
```

```
    Lauki:
        No vecāka klases Klienti:
            vards (str): Klienta vārds
            uzvards (str): Klienta uzvārds
            tips (str): Klienta tips
            telefons (str): Klienta kontakts
        organizacija (str): Juridiskas personas pārstāvēta organizācija

    Metodes:
        __str__()
    '''

    def __init__(self, vards, uzvards, tips, telefons, organizacija):
        '''
        Konstruktors.
        Piešķir sākotnējas vērtības laukiem.

        Argumenti:
            No vecāka klases Klienti:
                vards (str): Klienta vārds
                uzvards (str): Klienta uzvārds
                tips (str): Klienta tips
                telefons (str): Klienta kontakts
            organizacija (str): Juridiskas personas pārstāvēta organizācija
        '''

        super().__init__(vards, uzvards, tips, telefons)
        self.organizacija = organizacija

    def __str__(self):
        '''
        Metode.
        Izraksta visu informāciju par klientu.
        '''

        return f'Juridiska persona {self.vards} {self.uzvards}, tel.: {self.telefons}, pārstāvēta organizācija:
{self.organizacija}'


def galvena_programma():
```

```
    print('Ispējamas darbības: ')
    print('  --- pirkt auto (p)')
    print('  --- utilizēt auto (u)')
    print('  --- atdod auto nomai (n)')
    print('  --- saņemt auto atpakaļ (s)')
    print('  --- izvadīt visus pieejamus auto (a)')
    print('  --- izvadīt visus nomātos auto ar klientu informāciju (i)')
    print('  --- izvadīt nomas punkta kontaktus (k)')
    print('  --- beigt darbu (b)')

    n = Nomas_punkts([], [])

    while True:
        print(' ')
        darb = input('Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> ')
        print(' ')

        match darb:
            case 'p':
                n.pirkt()
            case 'u':
                n.utilizet()
            case 'n':
                n.iznomat()
            case 's':
                n.sanemt()
            case 'a':
                n.visi_auto()
            case 'i':
                n.iznomatie()
            case 'k':
                n.kontakti()
            case 'b':
                print('Programma beidz darbību.')
                break
            case _:
                print('Nav tādas darbības!')


if __name__ == '__main__':
    galvena_programma()
```

Testa piemērs(1)

```
Ispējamas darbības:
  --- pirkt auto (p)
  --- utilizēt auto (u)
  --- atdod auto nomai (n)
  --- saņemt auto atpakaļ (s)
  --- izvadīt visus pieejamus auto (a)
  --- izvadīt visus nomātos auto ar klientu informāciju (i)
  --- izvadīt nomas punkta kontaktus (k)
  --- beigt darbu (b)

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> p

Ievadiet auto marku --> Volvo
Ievadiet auto tipu (k - krāvas, v - vieglais, a - autobuss) --> v
Ievadiet ātrumkārbas tipu --> mehāniska
Ievadiet maksimālo pārvadājamo pasažieru skaitu --> 5

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> p

Ievadiet auto marku --> Seat
Ievadiet auto tipu (k - krāvas, v - vieglais, a - autobuss) --> v
Ievadiet ātrumkārbas tipu --> automatiska
Ievadiet maksimālo pārvadājamo pasažieru skaitu --> 7

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> a

Vieglais auto nr.1, ātrumkārba mehāniska, max pārvad. pasažieru skaits 5
Vieglais auto nr.2, ātrumkārba automatiska, max pārvad. pasažieru skaits 7
```

Testa piemērs(2)

```
Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> n

Ievadiet iznomātas mašīnas id --> 2
Ievadiet vārdu --> Simona
Ievadiet uzvārdu --> Blinova
Ievadiet telefona numuru --> 0000
Ievaidet personas tipu (f - fiziska, j - juridiska) --> f
Ievadiet vadītāja kategoriju (B - vieglie, C - krāvas, D - autobuss) --> B

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> i

Fiziska persona Simona Blinova, tel.: 0000, kategorija: B
Vieglais auto nr.2, ātrumkārba automatiska, max pārvad. pasažieru skaits 7

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> a

Vieglais auto nr.1, ātrumkārba mehāniska, max pārvad. pasažieru skaits 5
```

Testa piemērs(3)

```
Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> n

Ievadiet iznomātas mašīnas id --> 1
Ievadiet vārdu --> Simona
Ievadiet uzvārdu --> Blinova
Ievadiet telefona numuru --> 0000
Ievaidet personas tipu (f - fiziska, j - juridiska) --> f
Ievadiet vadītāja kategoriju (B - vieglie, C - krāvas, D - autobuss) --> B

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> a

Nomā nav auto!

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> i

Fiziska persona Simona Blinova, tel.: 0000, kategorija: B
Vieglais auto Audi nr.1, ātrumkārba automatiska, max pārvad. pasažieru skaits 5

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> s

Ievadiet saņēmamas auto id --> 1

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> a

Vieglais auto Audi nr.1, ātrumkārba automatiska, max pārvad. pasažieru skaits 5

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> i

Nav iznomāto auto!

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> k

Nomas punkta tel.: +37100000000, email: nomaspunkts@inbox.lv

Ko vēlaties darīt? (p, u, n, s, a, i, k, b) --> b

Programma beidz darbību.
```

2.uzdevums

Programma, kas nosaka dažādu figūru apkārtmēru un laukumu

Kods:

```
import math

class Figura:
  def __init__(self):
    self._pi = 3.14

  def perimetrs(self):
```

```
        pass

    def laukums(self):
        pass


class Taisnsturis(Figura):
    def __init__(self, garums, platums):
        self.garums = garums
        self.platums = platums

    def perimetrs(self):
        return (self.garums + self.platums) * 2

    def laukums(self):
        return self.garums * self.platums


class Rinkis(Figura):
    def __init__(self, radiuss):
        super().__init__()
        self.radiuss = radiuss

    def perimetrs(self):
        return 2 * self._pi * self.radiuss

    def laukums(self):
        return self._pi * self.radiuss ** 2


class Trijsturis(Figura):
    def __init__(self, mala1, mala2, mala3):
        self.mala1 = mala1
        self.mala2 = mala2
        self.mala3 = mala3

    def perimetrs(self):
        return self.mala1 + self.mala2 + self.mala3

    def laukums(self):
        pusp = self.perimetrs() / 2
```

```
        return math.sqrt(pusp * (pusp-self.mala1) * (pusp - self.mala2) * (pusp - self.mala3))


def izdrukat_informaciju(figura):
    print(f'Perimetrs: {figura.perimetrs()}')
    print(f'Laukums: {figura.laukums()}')


def galvena_programma():
    print('Iespējamas darbības:')
    print('  --- izveidot taisnstūri (t)')
    print('  --- izveidot riņķi (r)')
    print('  --- izveidot trijstūri (tr)')
    print('  --- beigt darbu (b)')

    while True:
        print(' ')
        darb = input('Ko vēlaties darīt? (t, r, tr, b) --> ')
        print(' ')

        match darb:
            case 't':
                g = int(input('Ievadiet taisnstūra garumu --> '))
                p = int(input('Ievadiet taisnstūra platumu --> '))

                if g < 1 or p < 1:
                    print('Tāda mala nevar būt!')
                else:
                    f = Taisnsturis(g, p)
                    izdrukat_informaciju(f)

            case 'r':
                r = int(input('Ievadiet riņķa līnijas radiusu --> '))

                if r < 1:
                    print('Tāds radiuss nevar būt!')
                else:
                    f = Rinkis(r)
                    izdrukat_informaciju(f)

            case 'tr':
```

```
            m1 = int(input('Ievadiet 1. malas garumu --> '))
            m2 = int(input('Ievadiet 2. malas garumu --> '))
            m3 = int(input('Ievadiet 3. malas garumu --> '))

            if m1 < 1 or m2 < 1 or m3 < 1:
                print('Tādas malas nevar būt!')
            elif m1 + m2 > m3 and m1 + m3 > m2 and m2 + m3 > m1:
                f = Trijsturis(m1, m2, m3)
                izdrukat_informaciju(f)
            else:
                print('Tāds trijstūris nevar būt!')

        case 'b':
            print('Programma beidz darbību.')
            exit()

        case _:
            print('Tādas darbības nav!')

if __name__ == '__main__':
    galvena_programma()
```

Testa piemērs(1)

```
Iespējamas darbības:
  --- izveidot taisnstūri (t)
  --- izveidot riņķi (r)
  --- izveidot trijstūri (tr)
  --- beigt darbu (b)

Ko vēlaties darīt? (t, r, tr, b) --> t

Ievadiet taisnstūra garumu --> 12
Ievadiet taisnstūra platumu --> 3
Perimetrs: 30
Laukums: 36

Ko vēlaties darīt? (t, r, tr, b) --> r

Ievadiet riņķa līnijas radiusu --> 5
Perimetrs: 31.400000000000002
Laukums: 78.5

Ko vēlaties darīt? (t, r, tr, b) --> tr

Ievadiet 1. malas garumu --> 3
Ievadiet 2. malas garumu --> 4
Ievadiet 3. malas garumu --> 5
Perimetrs: 12
Laukums: 6.0

Ko vēlaties darīt? (t, r, tr, b) --> b

Programma beidz darbību.
```

Testa piemērs(2)

```
Ko vēlaties darīt? (t, r, tr, b) --> r

Ievadiet riņķa līnijas radiusu --> 1
Perimetrs: 6.28
Laukums: 3.14

Ko vēlaties darīt? (t, r, tr, b) --> r

Ievadiet riņķa līnijas radiusu --> 2
Perimetrs: 12.56
Laukums: 12.56

Ko vēlaties darīt? (t, r, tr, b) --> r

Ievadiet riņķa līnijas radiusu --> 3
Perimetrs: 18.84
Laukums: 28.26

Ko vēlaties darīt? (t, r, tr, b) --> b

Programma beidz darbību.
```

Testa piemērs(3)

```
Ko vēlaties darīt? (t, r, tr, b) --> t

Ievadiet taisnstūra garumu --> 1
Ievadiet taisnstūra platumu --> 1
Perimetrs: 4
Laukums: 1

Ko vēlaties darīt? (t, r, tr, b) --> t

Ievadiet taisnstūra garumu --> 2
Ievadiet taisnstūra platumu --> 14
Perimetrs: 32
Laukums: 28

Ko vēlaties darīt? (t, r, tr, b) --> b

Programma beidz darbību.
```