**9. praktiskais darbs. 2. semestris**

**PU2. uzdevums**

Realizēt kuģīšu šaušanas spēli jeb spēli KARTUPELIS – 2 spēlētāji dators un lietotājs.

1. Līmenis abu spēlētāju kuģu izvietojumu ievada lietotājs un uz ekrāna redzamas tikai lietotāja gājieni datora “jūrā” un lietotāja “jūra ar kuģiem” un datora izdarītiem gājieniem. Katrs secīgi var izdarīt tikai vienu gājienu.
2. Līmenis – dators pats izvieto savus kuģus “jūrā”
3. Līmenis – dators šāvienus izdara plānveidīgi nevis uz labu laimi.

**Kods:**

# Programmas nosaukums: Kuģīšu šaušanas spēle. 2.LĪMENIS.

# Papilduzdevums 2. (1MPR09\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Realizēt kuģīšu šaušanas spēli jeb spēli KARTUPELIS – 2 spēlētāji dators un lietotājs.

# Realizēt kuģīšu šaušanas spēli jeb spēli KARTUPELIS – 2 spēlētāji dators un lietotājs.

# 1. Līmenis abu spēlētāju kuģu izvietojumu ievada lietotājs un uz ekrāna redzamas tikai lietotāja gājieni datora “jūrā” un lietotāja “jūra ar kuģiem” un datora izdarītiem gājieniem.

# Katrs secīgi var izdarīt tikai vienu gājienu.

# 2. Līmenis – dators pats izvieto savus kuģus “jūrā”

# 3. Līmenis – dators šāvienus izdara plānveidīgi nevis uz labu laimi.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

import random

"""

TIKA REALIZĒTS 2. LĪMENIS - dators pats izvieto savus kuģus “jūrā”. Katrs secīgi var izdarīt tikai vienu gājienu.

Uz ekrāna redzamas tikai lietotāja gājieni datora “jūrā” un lietotāja “jūra ar kuģiem” un datora izdarītiem gājieniem.

Dators trāpa nejauši, bet tikai tas vietas, kas vel nebīja trāpītas.

Dators nejauši izvieto savus kuģus (un tie kuģi būs novietoti pēc noteiktumiem bez pārklašanos).

Cilvēks var izvelētis vai novietot nejauši savus kuģus (un tie kuģi būs novietoti pēc noteiktumiem bez pārklašanos), vai pats novietot.

Kad novieto pats, tad katru reizi pārbauda vai tas pārklajās ar citu kuģi (vai to var vispār tā novietot). Ja nevār, tad mēģinām atkāl.

Dators uzvār tad, kad uz cilvēka (players\_ships) galda ir "X" skaitā 20. (count\_x(sv) funkcija noteic simbolu virknes "X" skaitu).

Cilvēks uzvār tad, kad uz viņa (player\_shots) galda ir "X" skaitā 20. (count\_x(sv) funkcija noteic simbolu virknes "X" skaitu).

20 tas ir skaitlis, ko iegūvam saskaitot visu kuģu garumus.

kuģi ir šādiem gārumiem

SSSS

SSS

SSS

SS

SS

SS

S

S

S

S

Ar "S" tiek norādīti kuģa daļas (vai pats kuģis)

"""

# globālie mainīgie kuģu izvietojumiem

ships\_arrangement = numpy.zeros((10, 10), dtype=int)

players\_ships = numpy.zeros((10, 10), dtype=int)

ship\_lengths = [4, 3, 3, 2, 2, 2, 1, 1, 1, 1]

player\_shots = numpy.zeros((10, 10), dtype=int)

hits\_and\_misses = numpy.zeros((10, 10), dtype=int)

computer\_shots = numpy.zeros((10, 10), dtype=int)

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

def find\_sum(saraksts): # Šājā gadījumā ir 20. Jā būtu vairāk kuģu, (vai citās kombinācijas), tad cits skaitlis. # find\_sum(ship\_lengths).

# saraksts - saraksts, kurā meklēsim to visu elementu summa (visi elementi ir skaitļi).

# Atgriež visu saraksta elementu summu.

# Ir nepieciešāma programmā, lai noteiktu cik krustiņu (X) ir nepieciešāms, lai būtu uz galda, lai tiek paziņots par uzvaru vai zaudējumu.

total\_sum = 0

for num in saraksts:

total\_sum = total\_sum + num

return total\_sum

def count\_x(sv):

# Skaitā cik reizes parādas X simbolu virknē un atgriež to paradīšanas skaitu

# sv - simbolu virkne (string)

sk = 0

for s in sv:

if str(s) == "X":

sk = sk + 1

return sk

def create\_random\_ships\_arrangement\_for\_computer(): # "create\_random\_ships\_arrangement\_for\_computer()", kas izveidos nejauši kuģu izvietojumu datoram.

# Nejaušam kuģu izvietojumam datoram

global ships\_arrangement # global ir izmantots datoram kuģu izvietojumam

# izveidot tukšu 10x10 masīvu

board = numpy.zeros((10, 10), dtype=int)

# definēt kuģus un to garumus

ships = [(4, 1), (3, 2), (2, 3), (1, 4)] # Tiek definēti četri kuģi ar to garumiem un skaitiem. (garums, cik ir tas tipa kuģu)

# Tiek veikta iterācija pāri katram kuģim ar to garumu un skaitu.

for ship in ships:

(length, count) = ship # Kortežs. Katrs kuģis ir kortežs ar garumu un to skaitu).

for i in range(count): # Novietojam tik reizes, cik ir count (pēc nosācijumiem cik var novietot kuģus.

placed = False # "placed" tiek definēts kā "False". (Nav pagaidām novietots).

while not placed: # Kamēr kuģis nav novietots pareizi, tiek veikts while cikls.

# nejauši novietot kuģus

direction = random.randint(0, 1) # vai nu 0 vai 1

if direction == 0: # horizontāli h

x = random.randint(0, 9 - length + 1)

y = random.randint(0, 9)

# pārbauda, vai kuģis pārklājas ar citu kuģi

parklajums = False # pēc noklusējuma domājam ka varam novietot (nepārklājas)

for j in range(max(0, y - 1), min(10, y + 2)):

for k in range(max(0, x - 1), min(10, x + length + 1)):

if board[j, k] == 1:

parklajums = True # tas nozīme kā nevarēsim novietot

if not parklajums: # Ja visos pārbaudījumos ir OK

board[y, x:x + length] = 1

placed = True # Novietots

else: # vertikāli

x = random.randint(0, 9)

y = random.randint(0, 9 - length + 1)

# pārbauda, vai kuģis pārklājas ar citu kuģi

parklajums = False # pēc noklusējuma domājam, ka varam novietot (nepārklājas)

for j in range(max(0, y - 1), min(10, y + length + 1)):

for k in range(max(0, x - 1), min(10, x + 2)):

if board[j, k] == 1:

parklajums = True # tas nozīme, ka nevarēsim novietot

if not parklajums: # Ja visos pārbaudījumos ir OK

board[y:y + length, x] = 1

placed = True # Novietots

# iestatīt globālo mainīgo (tas datoram)

ships\_arrangement = board

def create\_random\_ships\_arrangement\_for\_player():

# Funkcija nejaušam kuģu izvietojumam cilvēkam.

# Tas pats kā datoram, bet tikai cits globālais mainīgais

# global players\_ships - numpy 10x10 divdimensijas masīvs ar cilvēka kuģiem

global players\_ships

# izveidot tukšu 10x10 masīvu

board = numpy.zeros((10, 10), dtype=int)

# definēt kuģus un to garumus

ships = [(4, 1), (3, 2), (2, 3), (1, 4)]

for ship in ships: # cikls ar sarakstu no kortēžiem

(length, count) = ship

for i in range(count): # ejam cikla tik, cik ir to noteiktu kuģu (length, count)

placed = False

while not placed:

# nejauši novietot kuģus

direction = numpy.random.randint(2)

if direction == 0: # horizontāli

x = numpy.random.randint(0, 10 - length + 1)

y = numpy.random.randint(0, 10)

# pārbauda, vai kuģis pārklājas ar citu kuģi

parklajums = False # pieņēmam kā neparklājas (karogs), ja kaut vienā pārbaudījuma pārklājas, tad nevaram novietot

for j in range(max(0, y - 1), min(10, y + 2)):

for k in range(max(0, x - 1), min(10, x + length + 1)):

if board[j, k] == 1:

parklajums = True # ja neizpildās, tad pārklājas un nevaram novietot

if not parklajums:

board[y, x:x + length] = 1

placed = True

else: # vertikāli

x = numpy.random.randint(0, 10)

y = numpy.random.randint(0, 10 - length + 1)

# pārbauda, vai kuģis pārklājas ar citu kuģi

parklajums = False

for j in range(max(0, y - 1), min(10, y + length + 1)):

for k in range(max(0, x - 1), min(10, x + 2)):

if board[j, k] == 1:

parklajums = True

if not parklajums:

board[y:y + length, x] = 1

placed = True

# iestatīt globālo mainīgo (cilvēkam)

players\_ships = board

def player\_hits\_and\_misses():

# Izveido cilvēka (spēlētaju) glīto galda attēlojumu izmantojot string

# global player\_shots - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka gājienus (trāpijumus)

# global ships\_arrangement - 10x10 numpy divdimensijas masīvs, kurš reprezentē datora kuģu novietošanu (nejauši izveidotu)

# global hits\_and\_misses - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka kuģu novietošanu (tas varētu būt vai nejauši izveidots, vai manuāli)

global player\_shots, ships\_arrangement, hits\_and\_misses

# Atjaunināt hits\_and\_miss dēli ar spēlētāja trāpijumiem

for i in range(10):

for j in range(10):

if player\_shots[i, j] == 1 and ships\_arrangement[i, j] == 1:

hits\_and\_misses[i, j] = 1

elif player\_shots[i, j] == 1 and ships\_arrangement[i, j] == 0:

hits\_and\_misses[i, j] = -1

# Spēles galda izveidošanas virknes attēlojumā

board\_str = " A B C D E F G H I J\n"

for i in range(10):

if i != 9:

board\_str = board\_str + " " + str(i + 1) + " "

else:

board\_str = board\_str + str(i + 1) + " "

for j in range(10):

if hits\_and\_misses[i][j] == 1:

board\_str = board\_str + "X "

elif hits\_and\_misses[i][j] == -1:

board\_str = board\_str + "O "

else:

board\_str = board\_str + ". "

board\_str = board\_str + "\n"

return board\_str

def check\_overlap(players\_ships, x, y, length, direction):

# Pārbauda vai jaunais kuģis pārklājas ar kādu no eksistējošiem kuģiem uz galda. Funkcija cilvēkam.

# Atgriež True vai False. True - pārklājas (overlaps). False - neparklājas var novietot.

# players\_ships - 10x10 divdimensijas masīvs ar cilvēka kuģiem

# x - x koordināta (int skaitlis no 0 līdz 9)

# y - y koordināta (int skaitlis no 0 līdz 9)

# length - garums kuģiem

# direction - tas ir str "h" vai "H", vai "v", vai "V"

if direction == "h" or direction == "H": # horizontāls novietojums

if x + length > 10:

return True # True == nedēr, nevar šādi novietot

for i in range(x, x + length):

if players\_ships[y, i] == 1:

return True

if y > 0 and players\_ships[y - 1, i] == 1:

return True

if y < 9 and players\_ships[y + 1, i] == 1:

return True

if i > 0 and players\_ships[y, i - 1] == 1:

return True

if i < 9 and players\_ships[y, i + 1] == 1:

return True

if y > 0 and i > 0 and players\_ships[y - 1, i - 1] == 1:

return True

if y < 9 and i < 9 and players\_ships[y + 1, i + 1] == 1:

return True

if y > 0 and i < 9 and players\_ships[y - 1, i + 1] == 1:

return True

if y < 9 and i > 0 and players\_ships[y + 1, i - 1] == 1:

return True

elif direction == "v" or direction == "V": # vertikāls novietojums

if y + length > 10:

return True

for i in range(y, y + length):

if players\_ships[i, x] == 1:

return True

if x > 0 and players\_ships[i, x - 1] == 1:

return True

if x < 9 and players\_ships[i, x + 1] == 1:

return True

if i > 0 and players\_ships[i - 1, x] == 1:

return True

if i < 9 and players\_ships[i + 1, x] == 1:

return True

if x > 0 and i > 0 and players\_ships[i - 1, x - 1] == 1:

return True

if x < 9 and i < 9 and players\_ships[i + 1, x + 1] == 1:

return True

if x > 0 and i < 9 and players\_ships[i + 1, x - 1] == 1:

return True

if x < 9 and i > 0 and players\_ships[i - 1, x + 1] == 1:

return True

return False

def place\_player\_ships():

# Cilvēka (spēlētaja) kuģu novietošanai

random\_placement = input("Vai vēlaties nejauši novietot savus kuģus? (j - jā / n - nē): ") # Vai gribāt random novietošanu?

# Tas ir lai būtu lielāka izvēle starp j, ja, jā, Ja, Jā, JA, JĀ, n, ne, nē, Ne, Nē, NE, NĒ.

while (random\_placement != "j" and random\_placement != "n" and random\_placement != "J" and random\_placement != "jā"

and random\_placement != "ja" and random\_placement != "Jā" and random\_placement != "Ja" and random\_placement != "JA"

and random\_placement != "JĀ" and random\_placement != "N" and random\_placement != "ne" and random\_placement != "nē"

and random\_placement != "Ne" and random\_placement != "Nē" and random\_placement != "NE" and random\_placement != "NĒ"):

random\_placement = input("Kļūda! Vai vēlaties nejauši novietot savus kuģus? (j - jā / n - nē): ")

# Pārbaudam vai izvēle starp j, J, ja, jā, Ja, Jā, JA, JĀ tika izdarīta.

if (random\_placement == "j" or random\_placement == "J" or random\_placement == "jā" or random\_placement == "ja"

or random\_placement == "Jā" or random\_placement == "Ja" or random\_placement == "JA" or random\_placement == "JĀ"):

create\_random\_ships\_arrangement\_for\_player() # Ja tika izvelēts j, ja, jā, Ja, Jā, JA, JĀ (viens no tiem), tad izveidosim cilvēka kuģus nejauši

# Pārbaudam vai izvēle starp n, ne, nē, Ne, Nē, NE, NĒ tika izdarīta.

if (random\_placement == "n" or random\_placement == "N" or random\_placement == "ne" or random\_placement == "nē"

or random\_placement == "Ne" or random\_placement == "Nē" or random\_placement == "NE" or random\_placement == "NĒ"):

# Ja tika izvelēts n, N, ne, nē, Ne, Nē, NE, NĒ (viens no tiem), tad manuāli veidosim kuģu novietojumu un kātru reizi pārbaudīsim vai tur varam to novietot.

for length in ship\_lengths:

placed = False

while not placed:

print("Kuģa novietošana ar garumu", length)

x = get\_input\_x() # paprāsam x koordinātu (A, B, C, D, E, F G, H, I, J (0-9))

y = get\_input\_y()

direction = input("Ievadiet virzienu (h - horizontāli, v - vertikāli): ")

if direction == 'h' or direction == 'H':

if check\_overlap(players\_ships, x, y, length, direction):

print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")

continue

if not check\_overlap(players\_ships, x, y, length, direction):

players\_ships[y, x:x + length] = 1

print(players\_ships\_board())

placed = True

else:

print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")

elif direction == 'v' or direction == 'V':

if check\_overlap(players\_ships, x, y, length, direction):

print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")

continue

if not check\_overlap(players\_ships, x, y, length, direction):

players\_ships[y:y + length, x] = 1

print(players\_ships\_board())

placed = True

else:

print("Šis kuģis pārklājas ar citu kuģi! Lūdzu, izvēlieties citu atrašanās vietu!")

else:

print("Nederīgs virziens! Lūdzu, ievadiet “h” horizontālam kuģu novietojumam vai “v” vertikālam kuģu noveitojumam.")

def get\_input\_x():

# Paprasīt ievādīt x koordinātas, kāmer nav ievādītas no A līdz J str.

# Pārkonvertēt A-J kā skaitļi no 0 līdz 9.

# Atgriež skaitli num no 0 līdz 9 (num - intervālā 0 līdz 9)

while True:

try:

symbol = input("Ievadiet koordinātu A-J: ")

# Var arī izmantot match case.

# Neizmantoju match case, jo nevisas Python versijas tas strāda.

if symbol == "A" or symbol == "a":

num = 0

elif symbol == "B" or symbol == "b":

num = 1

elif symbol == "C" or symbol == "c":

num = 2

elif symbol == "D" or symbol == "d":

num = 3

elif symbol == "E" or symbol == "e":

num = 4

elif symbol == "F" or symbol == "f":

num = 5

elif symbol == "G" or symbol == "g":

num = 6

elif symbol == "H" or symbol == "h":

num = 7

elif symbol == "I" or symbol == "i":

num = 8

elif symbol == "J" or symbol == "j":

num = 9

else:

print("Nekorekta ievāde. Ievadiet koordinātas!")

raise ValueError

return num

except:

pass

def get\_input\_y():

# Paprasīt ievādīt y koordinātas, kāmer nav ievādītas no 1 līdz 10 int.

# Atgriež ievādīto skaitli -1 (num-1 intervālā 0 līdz 9)

while True:

try:

num = int(input("Ievadiet koordinātu (1-10): "))

num = num - 1

if num >= 0 and num <= 9:

return num

else:

print("Nekorekta ievāde. Ievadiet koordinātu (1-10): ")

except ValueError:

print("Nekorekta ievāde. Ievadiet koordinātu (1-10):")

def player\_turn():

# Cilvēka gājiens

# global ships\_arrangement - 10x10 numpy divdimensijas masīvs, kurš reprezentē datora kuģu novietošanu (nejauši izveidotu)

# global players\_ships - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka kuģu novietošanu (tas varētu būt vai nejauši izveidots, vai manuāli)

# global player\_shots - 10x10 numpy divdimensijas masīvs, kurš reprezentē cilvēka gājienus (trāpijumus)

global ships\_arrangement, players\_ships, player\_shots

# Paprāsit lietotājam ievādit koordinātas

x = get\_input\_x() # x koordināta

y = get\_input\_y() # y koordināta

# Pārbaudām vai tas trāpis vai garām un izmainām vērtības matricas (mainām vērtības globālajos divdimensijas masīviem)

if ships\_arrangement[y][x] == 1: # Ja ir kuģis šajā "šuna"

print("Jūs trāpijat datora kuģi!")

ships\_arrangement[y][x] = 1

player\_shots[y][x] = 1

else:

print("Garām!\n")

ships\_arrangement[y][x] = 0

player\_shots[y][x] = 1

print(player\_hits\_and\_misses()) # izvādīt galdiņu ar cilvēka gājieniem

def computer\_turn():

# Datora gājiens

# global players\_ships - numpy masīvs 10x10 ar cilvēka kuģiem

# computer\_shots - numpy masīvs 10x10 ar datora gājieniem

global players\_ships, computer\_shots

# paņemam nejaušus koordinātas kā kortēžu coords([x,y]), no 0 līdz 9. (programmā ir no 0 līdz 9, cilvēkam tas izvādīts kā no 1 līdz 10)

coords = tuple(random.randint(0, 9) for i in range(2))

while computer\_shots[coords[1]][coords[0]] == -1 or computer\_shots[coords[1]][coords[0]] == -2: # X vai O

coords = tuple(random.randint(0, 9) for i in range(2)) # kāmer nav izšauts tāda vietā, kur tas vēl nebīja

if coords[0] == 0: # var izmantot match case, bet ne visiem Python vērsijas tas strādā, tāpēc ar elif

symbol = "A"

elif coords[0] == 1: # symbols, lai izvadītu to glīti uz ekrāna

symbol = "B"

elif coords[0] == 2: # 0 -> A, 1 -> B, 2 -> C, 3 -> D, 4 -> E, 5 -> F, 6 -> G, 7 -> H, 8 -> I, 9 -> J

symbol = "C"

elif coords[0] == 3:

symbol = "D"

elif coords[0] == 4:

symbol = "E"

elif coords[0] == 5:

symbol = "F"

elif coords[0] == 6:

symbol = "G"

elif coords[0] == 7:

symbol = "H"

elif coords[0] == 8:

symbol = "I"

elif coords[0] == 9:

symbol = "J"

# Pārbaudam vai dators trāpija Jūsu kuģi vai nē un izvadam to un pamainam vērtības global mainīgajos.

if players\_ships[coords[1]][coords[0]] == 1: # ja ir kuģis tad tur ir 1

print("Dators trāpija Jūsu kuģi ar koordinātam (" + symbol + ", " + str(coords[1] + 1) + ")")

players\_ships[coords[1]][coords[0]] = -1

computer\_shots[coords[1]][coords[0]] = -1 # kad trāpija, tad -1

else:

print("Dators trāpija garām! Datora šaviena koordinātas (" + symbol + ", " + str(coords[1] + 1) + ")")

players\_ships[coords[1]][coords[0]] = -2

computer\_shots[coords[1]][coords[0]] = -2 # kad garām, tad -2

def players\_ships\_board():

# Glīti ar koordinātam izvāda cilvēka kuģus uz ekrāna

# Tikai atgriež to galdu, bet neprintē.

# Lai printētu vajag print(players\_ships\_board())

# global players\_ships - numpy masīvs 10x10 ar cilvēka kuģiem

global players\_ships

board = " A B C D E F G H I J \n" # koordinātas

# glītai izvadīšanai

k = 1

for row in players\_ships:

if k != 10:

board += " " + str(k) + " "

else:

board += str(k) + " "

for cell in row:

if cell == 1: # S - tur ir kuģis (kuģa daļa)

board += "S "

elif cell == -1: # X - trāpita kuģa daļa

board += "X "

elif cell == -2: # O - garām (nav trāpīts)

board += "O "

else:

board += ". " # punkts uz galda

k = k + 1

board += "\n"

return board

def play\_game():

# Funckija spēles sākumam (kad jau notiek trāpīšanas un gājieni)

# ships\_arrangement - 10x10 divdimensiju masīvs ar datora kuģiem

# players\_ships - 10x10 divdimensiju masīvs ar cilvēka kuģiem

# player\_shots - 10x10 divdimensiju masīvs ar cilvēka izdarītiem gājieniem

global ships\_arrangement, players\_ships, player\_shots

print("Jūsu kūģi:")

print(players\_ships\_board()) # Izvadīt cilvēka galdiņu ar kuģiem

while True:

player\_turn() # cilvēka gājiens

summa = find\_sum(ship\_lengths) # summa ir 20 ar šadiem noteikumiem (20 - cik ir X maksimāli, cik punktu būs novietoti kuģi - visu kuģu gārumu summa)

if count\_x(player\_hits\_and\_misses()) == summa: # Ja cilvēks trāpija visus 20 (parādas 20 X uz galda), tad cilvēks uzvarēja

# Testēšanai: Ja summa == 1, tad pēc pirmas trāpīšanas (ir viens X uz galda) paradīsies tas, ka cilvēks uzvarēja.

print("Jūs uzvarējat! Apsveicam!")

quit() # programma beidzās

print("Datora gājiens...")

computer\_turn() # procedūra datora gājienam

# pārbauda, vai visi spēlētāja (cilvēka) kuģi ir iznīcināti

if count\_x(players\_ships\_board()) == summa: # Testēšanai summu var pamainīt (tā ir 20 tagad).

# Testēšanai: Ja summa == 1, tad pēc pirmas trāpīšanas (ir viens X uz galda) paradīsies tas, ka dators uzvarēja.

print("")

print(players\_ships\_board()) # Izvadīt cilvēka galdiņu ar kuģiem

print("Jūs zaudējat! Dators uzvarēja!")

quit() # programma beidzās

# Izvadīt cilvēka galdiņu ar kuģiem

print("Jūsu kuģi:")

print(players\_ships\_board())

'''

# TESTIEM

def print\_random\_ships(): # Tika izmantota testēšanai. Print datora kuģus.

global ships\_arrangement

# drukāt kolonnas koordinātas

print(" A B C D E F G H I J")

# drukas "spēles galdiņu" uz ekrāna un vertikāļu koordinātas

for i in range(10):

if i != 9:

print(" " + str(i + 1) + " ", end="") # end="" nozīme kā nav pāreju uz jauno rindu pēc print("", end=""). Jo parasti print izdāra arī kā /n

else:

print(str(i + 1) + " ", end="")

for j in range(10):

if ships\_arrangement[i][j] == 1:

print("S ", end="")

else:

print(". ", end="")

print("")

'''

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

# Galvenā programmas daļa

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

'''

globālie mainīgie programmā

ships\_arrangement

players\_ships

ship\_lengths

player\_shots

hits\_and\_misses

computer\_shots

'''

create\_random\_ships\_arrangement\_for\_computer() # Funkcija nejaušam kuģu izvietojumam datoram

# print\_random\_ships() # Testēšanai (izvada datora kuģus)

place\_player\_ships() # Funkcija kuģu izvietojumam cilvēkam (jautās, vai nejauši izvietot, vai manuāli)

play\_game() # gājieni

**Testa piemēri:**

1. Testa piemēri tiek parādīti video: <https://youtu.be/lI3G3xCvc_o>
2. Testā piemērus skatīt failos testa\_piemērs\_1.txt, testa\_piemērs\_2.txt, testa\_piemērs\_3.txt, testa\_piemērs\_4.txt. Šeit nepieliku testa piemērus, jo tad tas aizņemtu pārāk daudz atmiņas un lappuses.

Text

Description automatically generated

3)

Text

Description automatically generated

4)

A picture containing text

Description automatically generated