**15. praktiskais darbs. 2. semestris**

**1. uzdevums**

Sastādīt programmu, kas skaita, cik un kādi latīņu alfabēta burti (lielie un mazie burti netiek šķiroti) satur dotais teksts. Rezultātu uz ekrāna izvada burtu biežuma dilstošā secībā.

**Kods:**

# Programmas nosaukums: Latiņu alfabētu burtu biežums tekstā

# 1. uzdevums (1MPR15\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas skaita, cik un kādi latīņu alfabēta burti (lielie un mazie burti netiek šķiroti) satur dotais teksts.

# Rezultātu uz ekrāna izvada burtu biežuma dilstošā secībā.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

class SymbolCount:

def \_\_init\_\_(self, burts, skaits):

# Inicializācija.

# Instancei ir burts un skaits vertības.

# Strāda lidzīgi kortēžam ("burts", skaits), piemēram ("A", 0).

self.burts = burts

self.skaits = skaits

def \_\_repr\_\_(self):

# Izvada lietotājam instances burtu un skaitu, izmantojot print.

# Piemēram:

# print(alphabet[0])

# Izvada:

# ("A", 24422)

return f'("{self.burts}", {self.skaits})'

@staticmethod

def create\_latin\_alphabet():

# Atgriež sarakstu ar lieliem latiņu alfabētu burtiem alfabētiska secība, kur katram burtam ir piekārtots skaits 0.

# Lidzīgi kā atgriež tādu sarakstu ar kortežiem:

# [("A", 0), ("B", 0), ("C", 0), ("D", 0), ("E", 0), ("F", 0), ("G", 0), ("H", 0), ("I", 0), ("J", 0), ("K", 0), ("L", 0), ("M", 0), ("N", 0), ("O", 0), ("P", 0), ("Q", 0), ("R", 0), ("S", 0), ("T", 0), ("U", 0), ("V", 0), ("W", 0), ("X", 0), ("Y", 0), ("Z", 0)]

# Izmanto chr funkciju.

saraksts = []

for i in range(26):

burts = chr(i + 65)

skaits = 0

saraksts.append(SymbolCount(burts, skaits))

return saraksts

@staticmethod

def update\_symbol\_count(alphabet, symbol):

# Atjauno (palielinā par vienu) simbolu (burtu) skaitu alphabet mainīgajā, konkrētāja vietā (kur ir tās noteikts kortežs ar atbilstošu burtu un skaitu).

# alphabet - saraksts, kas izveidots ar latiņu alfabēta burtiem alfabētiska secība, kur katram burtam ir piekārtots skaitlis, kas parāda to burta biežumu.

# alphabet tiek izveidots izmantojot SymbolCount.create\_latin\_alphabet()

# alphabet = SymbolCount.create\_latin\_alphabet()

# symbol - simbols, kurš ir uzrakstīts ar Unicode skaitļi. Izvelēsimies kādu simbolu skaitu atjaunosim.

symbol = symbol.upper()

kods = ord(symbol)

if 65 <= kods <= 90:

index = kods - 65

alphabet[index].skaits += 1

@staticmethod

def update\_symbol\_count\_for\_all\_text(alphabet, text):

# Atjauno simbolu (burtu) skaitu visam tekstam mainīgajā alphabet.

# text - simbolu virkne (str), kurai atjaunosim visu burtu biežumu vērtības.

# alphabet - saraksts, kas izveidots ar latiņu alfabēta burtiem alfabētiska secība, kur katram burtam ir piekārtots skaitlis, kas parāda to burta biežumu.

# alphabet tiek izveidots izmantojot SymbolCount.create\_latin\_alphabet()

# alphabet = SymbolCount.create\_latin\_alphabet()

for char in text:

SymbolCount.update\_symbol\_count(alphabet, char)

@staticmethod

def print\_symbols\_by\_frequency(alphabet):

# Izvada lietotājam alphabeta visus burtus dilstoša secība pēc burtu biežuma. Izmanto print.

# alphabet - saraksts, kas izveidots ar latiņu alfabēta burtiem alfabētiska secība, kur katram burtam ir piekārtots skaitlis, kas parāda to burta biežumu.

# alphabet tiek izveidots izmantojot SymbolCount.create\_latin\_alphabet()

# alphabet = SymbolCount.create\_latin\_alphabet()

# Atgriež None

# Izvada jau sakartotu alfabētu, piemēram šādi:

# E 10

# T 5

# A 4

# I 3

# N 3

# ... (utt)

sorted\_alphabet = SymbolCount.sort\_sella\_dilstosa(alphabet)

for obj in sorted\_alphabet:

print(obj.burts, obj.skaits)

@staticmethod

def sort\_sella\_dilstosa(a):

# Sakārto masīvu dilstošā secībā, izmantojot Šellas metodi (Shell sort)

# a - saraksts (masīvs).

# Atgriež sakartotu masīvu dilstošā secībā.

n = len(a)

solis = (3 \*\* math.floor(math.log(2 \* n + 1, 3)) - 1) // 2

while solis >= 1:

for k in range(0, solis):

for i in range(solis + k, n, solis):

if a[i - solis].skaits < a[i].skaits:

x = a[i]

j = i

while a[j - solis].skaits < x.skaits:

a[j] = a[j - solis]

j = j - solis

if j == k:

break

a[j] = x

solis = (solis - 1) // 3

return a

def save\_text\_from\_data\_by\_rows\_to\_variable(datne):

# Atgriež visu nolasītu tekstu no .txt datnes kā vienu str mainīgu.

# datne - datnes fails (piemēram, .txt fails).

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

a = ""

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

a = a + rinda

return a

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

# Galvenā programmas daļa

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

datne = "C:\\Users\\User\\Desktop\\liels\_teksts\_eng.txt" # datne - ceļš līdz failām. Jāraksta ceļu ar \\

text = save\_text\_from\_data\_by\_rows\_to\_variable(datne) # Nolasam tekstu no datnes kā str.

# Izveido sarakstu ar visiem latiņu alfabēta burtiem (īstenība ar visiem angļu valodas alfabēta burtiem),

# kur katram burtam izmantojot klasi SymbolCount tiek piekārtots "skaits".

# Ejam pa ciklu un "saraksts.append(SymbolCount(burts, skaits))".

# Izveido alfabēta sarakstu (caur klasi SymbolCount), kur katram burtam ir piekārtots reižu skaits, cik tas parādas tekstā.

# Tas mainīgais ir līdzīgs šadam sarakstam ar kortēžiem iekšā:

# [("A", 0), ("B", 0), ("C", 0), ... , ("Z", 0)]

alphabet = SymbolCount.create\_latin\_alphabet()

SymbolCount.update\_symbol\_count\_for\_all\_text(alphabet, text) # Atjauno simbolu skaitu (burtu skaitu) katram simbolam (burtam) alfabēta.

SymbolCount.print\_symbols\_by\_frequency(alphabet) # Izvada visus simbolus (burtus) dilstoša skaita pēc biežuma.

**Testa piemēri:**

A black background with white letters and numbers

Description automatically generated with low confidence

2)

E 24422

T 18887

A 16410

I 16194

N 15128

O 14588

R 13986

S 13621

D 8507

H 8114

L 7859

C 7213

U 5955

M 5647

P 4933

F 4329

G 3880

Y 2974

W 2844

B 2828

V 2444

K 1165

J 468

X 380

Z 216

Q 141

1. Tekstu avoti ir pieminēti .txt failā.

.txt fails:

liels\_teksts\_eng.txt

A screenshot of a computer

Description automatically generated with low confidence

**PU1. uzdevums**

Sastādīt programmu, kas skaita, cik un kādi latviešu alfabēta burti (lielie un mazie burti netiek šķirti) satur dotais teksts. Rezultātu uz ekrāna izvada burtu biežuma dilstošā secībā.

**Kods:**

# Programmas nosaukums: Latviešu alfabētu burtu biežums tekstā

# PU1. uzdevums (1MPR15\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas skaita, cik un kādi latviešu alfabēta burti (lielie un mazie burti netiek šķirti) satur dotais teksts.

# Rezultātu uz ekrāna izvada burtu biežuma dilstošā secībā.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import math

class SymbolCountLv:

def \_\_init\_\_(self, burts, skaits):

# Inicializācija.

# Instancei ir burts un skaits vērtības.

# Strāda lidzīgi kortēžam ("burts", skaits), piemēram ("A", 0).

self.burts = burts

self.skaits = skaits

def \_\_repr\_\_(self):

# Izvada lietotājam instances burtu un skaitu, izmantojot print.

# Piemēram:

# print(alphabet[0])

# Izvada:

# ("A", 972)

return f'("{self.burts}", {self.skaits})'

@staticmethod

def create\_latvian\_alphabet():

# Atgriež sarakstu ar lieliem latviešu alfabētu burtiem alfabētiska secība, kur katram burtam ir piekārtots skaits 0.

# Lidzīgi kā atgriež tādu sarakstu ar kortežiem:

# [("A", 0), ("Ā", 0), ("B", 0), ("C", 0), ("Č", 0), ("D", 0), ("E", 0), ("Ē", 0), ("F", 0), ("G", 0), ("Ģ", 0), ("H", 0), ("I", 0), ("Ī", 0), ("J", 0), ("K", 0), ("L", 0), ("Ļ", 0), ("M", 0), ("N", 0), ("Ņ", 0), ("O", 0), ("P", 0), ("R", 0), ("S", 0), ("Š", 0), ("T", 0), ("U", 0), ("V", 0), ("Z", 0), ("Ž", 0)]

# Izmanto chr funkciju.

mas = []

lv = ["A", "Ā", "B", "C", "Č", "D", "E", "Ē", "F", "G", "Ģ", "H", "I", "Ī", "J", "K", "Ķ", "L", "Ļ", "M", "N", "Ņ", "O", "P", "R", "S", "Š", "T", "U", "Ū", "V", "Z", "Ž"]

for i in range(len(lv)):

burts = lv[i]

skaits = 0

mas.append(SymbolCountLv(burts, skaits))

return mas

@staticmethod

def update\_symbol\_count(alphabet, x):

# Atjauno (palielinā par vienu) simbolu (burtu) skaitu alphabet mainīgajā, konkrētāja vietā (kur ir tās noteikts kortežs ar atbilstošu burtu un skaitu).

# alphabet - saraksts, kas izveidots ar latiņu alfabēta burtiem alfabētiska secība, kur katram burtam ir piekārtots skaitlis, kas parāda to burta biežumu.

# alphabet tiek izveidots izmantojot SymbolCountLv.create\_latvian\_alphabet()

# alphabet = SymbolCountLv.create\_latvian\_alphabet()

# symbol - simbols, kurš ir uzrakstīts ar Unicode skaitļi. Izvelēsimies kādu simbolu skaitu atjaunosim.

lv = ["A", "Ā", "B", "C", "Č", "D", "E", "Ē", "F", "G", "Ģ", "H", "I", "Ī", "J", "K", "Ķ", "L", "Ļ", "M", "N", "Ņ", "O", "P", "R", "S", "Š", "T", "U", "Ū", "V", "Z", "Ž"]

x = x.upper()

if x in lv:

index = lv.index(x)

alphabet[index].skaits += 1

@staticmethod

def update\_symbol\_count\_for\_all\_text(alphabet, text):

# Atjauno simbolu (burtu) skaitu visam tekstam mainīgajā alphabet.

# text - simbolu virkne (str), kurai atjaunosim visu burtu biežumu vērtības.

# alphabet - saraksts, kas izveidots ar latiņu alfabēta burtiem alfabētiska secība, kur katram burtam ir piekārtots skaitlis, kas parāda to burta biežumu.

# alphabet tiek izveidots izmantojot SymbolCount.create\_latin\_alphabet()

# alphabet = SymbolCount.create\_latin\_alphabet()

for char in text:

SymbolCountLv.update\_symbol\_count(alphabet, char)

@staticmethod

def print\_symbols\_by\_frequency(alphabet):

# Izvada lietotājam alphabeta visus burtus dilstoša secība pēc burtu biežuma. Izmanto print.

# alphabet - saraksts, kas izveidots ar latviešu alfabēta burtiem alfabētiska secība, kur katram burtam ir piekārtots skaitlis, kas parāda to burta biežumu.

# alphabet tiek izveidots izmantojot SymbolCountLv.create\_latvian\_alphabet()

# alphabet = SymbolCountLv.create\_latvian\_alphabet()

# Atgriež None

# Izvada jau sakartotu alfabētu, piemēram šādi:

# L 2

# Ļ 2

# B 1

# A 1

# I 0

# ... (utt)

sorted\_alphabet = SymbolCountLv.sort\_sella\_dilstosa(alphabet)

for obj in sorted\_alphabet:

print(obj.burts, obj.skaits)

@staticmethod

def sort\_sella\_dilstosa(a):

# Sakārto masīvu dilstošā secībā, izmantojot Šellas metodi (Shell sort)

# a - saraksts (masīvs).

# Atgriež sakartotu masīvu dilstošā secībā.

n = len(a)

solis = (3 \*\* math.floor(math.log(2 \* n + 1, 3)) - 1) // 2

while solis >= 1:

for k in range(0, solis):

for i in range(solis + k, n, solis):

if a[i - solis].skaits < a[i].skaits:

x = a[i]

j = i

while a[j - solis].skaits < x.skaits:

a[j] = a[j - solis]

j = j - solis

if j == k:

break

a[j] = x

solis = (solis - 1) // 3

return a

def save\_text\_from\_data\_by\_rows\_to\_variable(datne):

# Uzrakstā termināla lietotājam visu tekstu no .txt failā pa rindam.

# datne - datnes fails (piemēram, .txt fails)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

a = ""

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

a = a + rinda

return a

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

# Galvenā programmas daļa

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

datne = "C:\\Users\\User\\Desktop\\liels\_teksts\_lv.txt" # datne - ceļš līdz failām.

text = save\_text\_from\_data\_by\_rows\_to\_variable(datne) # Nolasam tekstu no datnes kā str.

# Izveido sarakstu ar visiem latviešu alfabēta burtiem, kur katram burtam, izmantojot klasi SymbolCountLv, tiek piekārtots "skaits".

# Ejam pa ciklu un "saraksts.append(SymbolCountLv(burts, skaits))".

# Izveido alfabēta sarakstu (caur klasi SymbolCountLv), kur katram burtam ir piekārtots reižu skaits, cik tas parādas tekstā.

# Tas mainīgais ir līdzīgs šadam sarakstam ar kortēžiem iekšā:

# [("A", 0), ("Ā", 0), ("B", 0), ("C", 0), ("Č", 0), ("D", 0), ... , ("Z", 0), ("Ž", 0)]

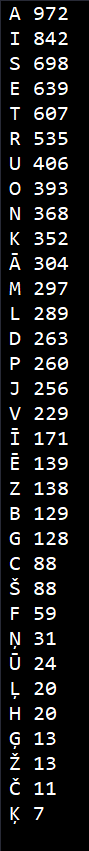
alphabet = SymbolCountLv.create\_latvian\_alphabet()

SymbolCountLv.update\_symbol\_count\_for\_all\_text(alphabet, text) # Atjauno simbolu skaitu (burtu skaitu) katram simbolam (burtam) alfabētā.

SymbolCountLv.print\_symbols\_by\_frequency(alphabet) # Izvada visus simbolus (burtus) dilstoša skaita pēc biežuma.

**Testa piemēri:**

1)



2)

A 972

I 842

S 698

E 639

T 607

R 535

U 406

O 393

N 368

K 352

Ā 304

M 297

L 289

D 263

P 260

J 256

V 229

Ī 171

Ē 139

Z 138

B 129

G 128

C 88

Š 88

F 59

Ņ 31

Ū 24

Ļ 20

H 20

Ģ 13

Ž 13

Č 11

Ķ 7

3) Tekstu avoti ir pieminēti .txt failā.

.txt fails:

liels\_teksts\_lv.txt

A screenshot of a computer

Description automatically generated with medium confidence

**2. uzdevums**

Sastādīt programmu, kas veic teksta šifrēšanu un atšifrēšanu, izmantojot šādu algoritmu.

# JĀNIS\_IR\_TEICAMNIEKS

# 19651965196519651965

# KĢŠLŠFĻUABILČGSSĪKŅV

**Kods:**

# Programmas nosaukums: Teksta šifrēšana (Latiņu burti)

# 2. uzdevums (1MPR15\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas veic teksta šifrēšanu un atšifrēšanu, izmantojot šādu algoritmu.

# JĀNIS\_IR\_TEICAMNIEKS

# 19651965196519651965

# KĢŠLŠFĻUABILČGSSĪKŅV

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

def write\_text\_to\_file(filename, text):

# NODZES VISU INFORMĀCIJU filename DATNE un ieraksta jaunu informāciju no str text mainīga.

# text - str teksts, kuru gribam ierākstit datnē.

# filename - faila (datnes) nosaukums.

# Piemēram:

# filename = "C:\\Users\\User\\Desktop\\nav\_sifrets\_1.txt"

with open(filename, mode='w', encoding='utf-8') as file:

file.write(text)

def save\_text\_from\_data\_by\_rows\_to\_variable(datne):

# Atgriež visu nolasītu tekstu no .txt datnes kā vienu str mainīgu.

# datne - datnes fails (piemēram, .txt fails).

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

a = ""

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

a = a + rinda

return a

def encrypt(text\_non\_sifrets, atslega, burti):

# Atgriež aizifrētu (sifrets) str tekstu, pamatojoties uz text\_non\_sifrets, atslega un burti.

# Tas ej ciklā (pa indeksiem) pa katru burtu text\_non\_sifrets uz atslega skaitu un tāda veida pa burtiem izveido jau aizšifrētu (sifrets) tekstu.

# Ej pa indeksiem uz priekšu uz atslega skaitu.

# text\_non\_sifrets - str teksts, kas nav šifrēts.

# atslega - simbolu virkne (str), kas sastāv no cipariem no 0 līdz 9 un tas nav lielāka nekā text\_sifrets.

# burti - saraksts ar visiem alfabēta burtiem, pec kuriem gribat aizšifrēt.

# Piemēram:

# burti = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "\_"]

sifrets = "" # Tukšais str, kuru piepildīsim ar aizšifrētiem burtiem.

for i in range(len(text\_non\_sifrets)): # Ej ciklā pa text\_non\_sifrets visiem burtiem.

char = text\_non\_sifrets[i] # char - i-tajs burts text\_non\_sifrets str tekstā.

if char in burti: # Pārbaudam vai rakstzīme (simbols) ir in burti list.

key = int(atslega[i % len(atslega)]) # Iegūstam atbilstošo atslēgas ciparu, izmantojot moduļ (%) operatoru.

encrypted\_char\_index = (burti.index(char) + key) % len(burti) # Aprēķinam aizšifrēto rakstzīmju indeksu.

encrypted\_char = burti[encrypted\_char\_index] # Iegūstam aizšifrētu rakstzīmi.

sifrets += encrypted\_char # Pievienojam aizšifrētu burtu simbolu virknei sifrets.

else:

sifrets += char # Pievienojam rakstzīmi tādu, kāds tas ir, ja tas nav in burti list.

return sifrets # Atgriež str simbolu virkni sifrets.

def decrypt(text\_sifrets, atslega, burti):

# Atgriež atšifrētu (nav\_sifrets) str tekstu, pamatojoties uz text\_sifrets, atslega un burti.

# Tas ej ciklā (pa indeksiem) pa katru burtu text\_sifreta uz atslega skaitu un tāda veida pa burtiem izveido jau atšifrētu (nav\_sifrets) tekstu.

# Ej pa indeksiem atpakaļ uz atslega skaitu.

# text\_sifrets - str teksts, kas ir šifrēts.

# atslega - simbolu virkne (str), kas sastāv no cipariem no 0 līdz 9 un tas nav lielāka nekā text\_sifrets.

# burti - saraksts ar visiem alfabēta burtiem, pec kuriem gribat atšifrēt.

# Piemēram:

# burti = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "\_"]

nav\_sifrets = "" # Tukšais str, kuru piepildīsim ar atšifrētiem burtiem.

for i in range(len(text\_sifrets)): # Ej ciklā pa text\_sifrets visiem burtiem.

char = text\_sifrets[i] # char - i-tajs burts text\_sifrets str tekstā.

if char in burti: # Pārbaudam vai rakstzīme (simbols) ir in burti list.

key = int(atslega[i % len(atslega)]) # Iegūstam atbilstošo atslēgas ciparu, izmantojot modules (%) operatoru.

decrypted\_char\_index = (burti.index(char) - key) % len(burti) # Aprēķinam atšifrēto rakstzīmju indeksu.

decrypted\_char = burti[decrypted\_char\_index] # Iegūstam atšifrētu rakstzīmi.

nav\_sifrets += decrypted\_char # Pievienojam atšifrētu burtu simbolu virknei nav\_sifrets.

else: # Ja burts nav in burti list, tad vienkarši pievienojam to rakstzīmi nav\_sifrets simbolu virknei.

nav\_sifrets += char # Pievienojam to pašu burtu simbolu virknei nav\_sifrets (ja tas burts nav in burti list)

return nav\_sifrets # Atgriež str simbolu virkni nav\_sifrets.

def input\_cypher\_or\_decrypt():

# Prasa lietotājam vai lietotājs grib aizšifrēt (encrypt) tekstu vai atšifrēt (decrypt) tekstu.

# Ja lietotājs ievādīs "c" vai "C", tad viņš grib aizšifrēt (encrypt).

# Ja lietotājs ievādīs "d" vai "D", tad viņš grib atšifrēt (decrypt).

# Atgriež True, ja ir ievādīts "c" vai "C" (str).

# Atgriež False, ja ir ievādīts "d" vai "D" (str).

cypher\_or\_decrypt = ""

while cypher\_or\_decrypt.lower() != "c" and cypher\_or\_decrypt.lower() != "d":

cypher\_or\_decrypt = input("Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> ")

if cypher\_or\_decrypt.lower() == "c":

return True

elif cypher\_or\_decrypt.lower() == "d":

return False

def input\_atslega(text\_non\_sifrets):

# Prasa lietotājam ievādīt atslēgu kāmer tas nav lielāka par text\_non\_sifrets vai kāmēr tā tav simbolu virkne bez cipariem.

# text\_non\_sifrets - str teksts, kas nav šifrēts (kuru gribat aizšifrēt).

# Atgriež atslega, kuru ievada lietotājs.

atslega = input("Ievadiet atslēgu ==> ")

while len(atslega) > len(text\_non\_sifrets) or not atslega.isdigit():

print("Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!")

atslega = input("Ievadiet atslēgu ==> ")

return atslega

def cypher\_main():

# Encryption.

# Ieraksta failā ir\_sifrets\_txt = "C:\\Users\\User\\Desktop\\ir\_sifrets\_1.txt" aizšifrētu tekstu.

# Ieraksta failā atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_1.txt" atslēgu, kuru ievādīs lietotājs, izmantojot input\_atslega funkciju.

text\_non\_sifrets = save\_text\_from\_data\_by\_rows\_to\_variable(nav\_sifrets\_txt) # Saglabam mainīgajā text\_non\_sifrets no nav\_sifrets\_txt datnes. Saglabam kā str virkni.

text\_non\_sifrets = text\_non\_sifrets.upper() # Visu str simbolus pārveidojam par lieliem burtiem.

print("\nNav šifrēts teksts:") # Informācija lietotājam par to, kads tagad teksts nav šifrēts (kuru mes aizšifrēsim pēc atslēgas).

print(text\_non\_sifrets) # Izvadam pagaidam nav aizšifrētu tekstu, ka simbolu virkni.

print() # Atstarpe glītumam.

atslega = input\_atslega(text\_non\_sifrets) # Prasam lietotājam ievādīt atslegu.

sifrets = encrypt(text\_non\_sifrets, atslega, burti) # Aizšifrējam sifrets simbolu virkni.

print()

print(f"Teksts tika aizšifrēts ar atslēgu {atslega}:")

print(sifrets) # Izvadam aizšifrētu tekstu lietotājam.

write\_text\_to\_file(ir\_sifrets\_txt, sifrets) # Ierakstam failā ir\_sifrets\_txt = "C:\\Users\\User\\Desktop\\ir\_sifrets\_1.txt" aizšifrētu tekstu.

write\_text\_to\_file(atslega\_txt, atslega) # Ierakstam failā atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_1.txt" atslegas kodu.

def decrypt\_main():

# Decryption.

# Ieraksta failā nav\_sifrets\_txt = "C:\\Users\\User\\Desktop\\nav\_sifrets\_1.txt" atšifrētu tekstu.

# Ieraksta failā atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_1.txt" atslēgu, kuru ievādīs lietotājs, izmantojot input\_atslega funkciju.

text\_sifrets = save\_text\_from\_data\_by\_rows\_to\_variable(ir\_sifrets\_txt) # Saglabam mainīgajā text\_sifrets no ir\_sifrets\_txt datnes. Saglabam kā str virkni.

text\_sifrets = text\_sifrets.upper() # Visu str simbolus pārveidojam par lieliem burtiem.

print("\nŠifrēts teksts:") # Informācija lietotājam par to, kā aizskatās šifrēts teksts (kuru mes atšifrēsim pēc atslēgas).

print(text\_sifrets) # Izvadam pagaidam nav atšifrētu tekstu, ka simbolu virkni.

print() # Atstarpe glītumam.

atslega = input\_atslega(text\_sifrets) # Prasam lietotājam ievādīt atslegu.

nav\_sifrets = decrypt(text\_sifrets, atslega, burti) # Atšifrējam nav\_sifrets simbolu virkni.

print()

print(f"Teksts tika atšifrēts ar atslēgu {atslega}:")

print(nav\_sifrets) # Izvadam atšifrētu tekstu lietotājam.

write\_text\_to\_file(nav\_sifrets\_txt, nav\_sifrets) # Ierakstam failā nav\_sifrets\_txt = "C:\\Users\\User\\Desktop\\nav\_sifrets\_1.txt" atšifrētu tekstu.

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

# Galvenā programmas daļa

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

nav\_sifrets\_txt = "C:\\Users\\User\\Desktop\\nav\_sifrets\_1.txt" # Šajā vietā lietotājs ievadīs tekstu kas nav šifrēts.

ir\_sifrets\_txt = "C:\\Users\\User\\Desktop\\ir\_sifrets\_1.txt" # Šajā vietā būs aizšifrēts teksts.

atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_1.txt" # Šajā vietā glabāsies atslega no aizšifrētam tekstam.

burti = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "\_"] # Latiņu alfabēts ar svitriņu.

# ievade - boolean vertība, ja True, tad encryption (atšifrēts -> aizšifrēts). Ja False, tad decryption (aizšifrēts -> atšifrēts)

ievade = input\_cypher\_or\_decrypt() # Prasa lietotājam vai lietotājs grib aizšifrēt (encrypt) tekstu vai atšifrēt (decrypt) tekstu.

if ievade: # Ja ievade ir True, tad lietotājs grib aizšifrēt tekstu.

cypher\_main()

elif not ievade: # Ja ievade ir False, tad lietotājs grib atšifrēt tekstu.

decrypt\_main()

else:

print("Neparedzamā kļūda!")

**Testa piemēri:**

1)

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

2)

A computer screen with white text

Description automatically generated with low confidence

A picture containing text, screenshot, font

Description automatically generated

3)

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

4)

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

**PU2. uzdevums**

Sastādīt programmu, kas veic teksta, kas sastāv no latviešu alfabēta burtiem, aizšifrēšanu un atšifrēšanu, atbilstoši 2.uzdevuma nosacījumiem.

**Kods:**

# Programmas nosaukums: Teksta šifrēšana (Latviešu burti)

# PU2. uzdevums (1MPR15\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas veic teksta, kas sastāv no latviešu alfabēta burtiem, aizšifrēšanu un atšifrēšanu,

# atbilstoši 2.uzdevuma nosacījumiem.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

def write\_text\_to\_file(filename, text):

# NODZES VISU INFORMĀCIJU filename DATNE un ieraksta jaunu informāciju no str text mainīga.

# text - str teksts, kuru gribam ierākstit datnē.

# filename - faila (datnes) nosaukums.

# Piemēram:

# filename = "C:\\Users\\User\\Desktop\\nav\_sifrets\_2.txt"

with open(filename, mode='w', encoding='utf-8') as file:

file.write(text)

def save\_text\_from\_data\_by\_rows\_to\_variable(datne):

# Atgriež visu nolasītu tekstu no .txt datnes kā vienu str mainīgu.

# datne - datnes fails (piemēram, .txt fails).

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

a = ""

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

a = a + rinda

return a

def encrypt(text\_non\_sifrets, atslega, burti):

# Atgriež aizifrētu (sifrets) str tekstu, pamatojoties uz text\_non\_sifrets, atslega un burti.

# Tas ej ciklā (pa indeksiem) pa katru burtu text\_non\_sifrets uz atslega skaitu un tāda veida pa burtiem izveido jau aizšifrētu (sifrets) tekstu.

# Ej pa indeksiem uz priekšu uz atslega skaitu.

# text\_non\_sifrets - str teksts, kas nav šifrēts.

# atslega - simbolu virkne (str), kas sastāv no cipariem no 0 līdz 9 un tas nav lielāka nekā text\_sifrets.

# burti - saraksts ar visiem alfabēta burtiem, pec kuriem gribat aizšifrēt.

# Piemēram:

# burti = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "\_"]

sifrets = "" # Tukšais str, kuru piepildīsim ar aizšifrētiem burtiem.

for i in range(len(text\_non\_sifrets)): # Ej ciklā pa text\_non\_sifrets visiem burtiem.

char = text\_non\_sifrets[i] # char - i-tajs burts text\_non\_sifrets str tekstā.

if char in burti: # Pārbaudam vai rakstzīme (simbols) ir in burti list.

key = int(atslega[i % len(atslega)]) # Iegūstam atbilstošo atslēgas ciparu, izmantojot moduļ (%) operatoru.

encrypted\_char\_index = (burti.index(char) + key) % len(burti) # Aprēķinam aizšifrēto rakstzīmju indeksu.

encrypted\_char = burti[encrypted\_char\_index] # Iegūstam aizšifrētu rakstzīmi.

sifrets += encrypted\_char # Pievienojam aizšifrētu burtu simbolu virknei sifrets.

else:

sifrets += char # Pievienojam rakstzīmi tādu, kāds tas ir, ja tas nav in burti list.

return sifrets # Atgriež str simbolu virkni sifrets.

def decrypt(text\_sifrets, atslega, burti):

# Atgriež atšifrētu (nav\_sifrets) str tekstu, pamatojoties uz text\_sifrets, atslega un burti.

# Tas ej ciklā (pa indeksiem) pa katru burtu text\_sifreta uz atslega skaitu un tāda veida pa burtiem izveido jau atšifrētu (nav\_sifrets) tekstu.

# Ej pa indeksiem atpakaļ uz atslega skaitu.

# text\_sifrets - str teksts, kas ir šifrēts.

# atslega - simbolu virkne (str), kas sastāv no cipariem no 0 līdz 9 un tas nav lielāka nekā text\_sifrets.

# burti - saraksts ar visiem alfabēta burtiem, pec kuriem gribat atšifrēt.

# Piemēram:

# burti = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "\_"]

nav\_sifrets = "" # Tukšais str, kuru piepildīsim ar atšifrētiem burtiem.

for i in range(len(text\_sifrets)): # Ej ciklā pa text\_sifrets visiem burtiem.

char = text\_sifrets[i] # char - i-tajs burts text\_sifrets str tekstā.

if char in burti: # Pārbaudam vai rakstzīme (simbols) ir in burti list.

key = int(atslega[i % len(atslega)]) # Iegūstam atbilstošo atslēgas ciparu, izmantojot modules (%) operatoru.

decrypted\_char\_index = (burti.index(char) - key) % len(burti) # Aprēķinam atšifrēto rakstzīmju indeksu.

decrypted\_char = burti[decrypted\_char\_index] # Iegūstam atšifrētu rakstzīmi.

nav\_sifrets += decrypted\_char # Pievienojam atšifrētu burtu simbolu virknei nav\_sifrets.

else: # Ja burts nav in burti list, tad vienkarši pievienojam to rakstzīmi nav\_sifrets simbolu virknei.

nav\_sifrets += char # Pievienojam to pašu burtu simbolu virknei nav\_sifrets (ja tas burts nav in burti list)

return nav\_sifrets # Atgriež str simbolu virkni nav\_sifrets.

def input\_cypher\_or\_decrypt():

# Prasa lietotājam vai lietotājs grib aizšifrēt (encrypt) tekstu vai atšifrēt (decrypt) tekstu.

# Ja lietotājs ievādīs "c" vai "C", tad viņš grib aizšifrēt (encrypt).

# Ja lietotājs ievādīs "d" vai "D", tad viņš grib atšifrēt (decrypt).

# Atgriež True, ja ir ievādīts "c" vai "C" (str).

# Atgriež False, ja ir ievādīts "d" vai "D" (str).

cypher\_or\_decrypt = ""

while cypher\_or\_decrypt.lower() != "c" and cypher\_or\_decrypt.lower() != "d":

cypher\_or\_decrypt = input("Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> ")

if cypher\_or\_decrypt.lower() == "c":

return True

elif cypher\_or\_decrypt.lower() == "d":

return False

def input\_atslega(text\_non\_sifrets):

# Prasa lietotājam ievādīt atslēgu kāmer tas nav lielāka par text\_non\_sifrets vai kāmēr tā tav simbolu virkne bez cipariem.

# text\_non\_sifrets - str teksts, kas nav šifrēts (kuru gribat aizšifrēt).

# Atgriež atslega, kuru ievada lietotājs.

atslega = input("Ievadiet atslēgu ==> ")

while len(atslega) > len(text\_non\_sifrets) or not atslega.isdigit():

print("Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!")

atslega = input("Ievadiet atslēgu ==> ")

return atslega

def cypher\_main():

# Encryption.

# Ieraksta failā ir\_sifrets\_txt = "C:\\Users\\User\\Desktop\\ir\_sifrets\_2.txt" aizšifrētu tekstu.

# Ieraksta failā atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_2.txt" atslēgu, kuru ievādīs lietotājs, izmantojot input\_atslega funkciju.

text\_non\_sifrets = save\_text\_from\_data\_by\_rows\_to\_variable(nav\_sifrets\_txt) # Saglabam mainīgajā text\_non\_sifrets no nav\_sifrets\_txt datnes. Saglabam kā str virkni.

text\_non\_sifrets = text\_non\_sifrets.upper() # Visu str simbolus pārveidojam par lieliem burtiem.

print("\nNav šifrēts teksts:") # Informācija lietotājam par to, kads tagad teksts nav šifrēts (kuru mes aizšifrēsim pēc atslēgas).

print(text\_non\_sifrets) # Izvadam pagaidam nav aizšifrētu tekstu, ka simbolu virkni.

print() # Atstarpe glītumam.

atslega = input\_atslega(text\_non\_sifrets) # Prasam lietotājam ievādīt atslegu.

sifrets = encrypt(text\_non\_sifrets, atslega, burti) # Aizšifrējam sifrets simbolu virkni.

print()

print(f"Teksts tika aizšifrēts ar atslēgu {atslega}:")

print(sifrets) # Izvadam aizšifrētu tekstu lietotājam.

write\_text\_to\_file(ir\_sifrets\_txt, sifrets) # Ierakstam failā ir\_sifrets\_txt = "C:\\Users\\User\\Desktop\\ir\_sifrets\_1.txt" aizšifrētu tekstu.

write\_text\_to\_file(atslega\_txt, atslega) # Ierakstam failā atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_1.txt" atslegas kodu.

def decrypt\_main():

# Decryption.

# Ieraksta failā nav\_sifrets\_txt = "C:\\Users\\User\\Desktop\\nav\_sifrets\_2.txt" atšifrētu tekstu.

# Ieraksta failā atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_2.txt" atslēgu, kuru ievādīs lietotājs, izmantojot input\_atslega funkciju.

text\_sifrets = save\_text\_from\_data\_by\_rows\_to\_variable(ir\_sifrets\_txt) # Saglabam mainīgajā text\_sifrets no ir\_sifrets\_txt datnes. Saglabam kā str virkni.

text\_sifrets = text\_sifrets.upper() # Visu str simbolus pārveidojam par lieliem burtiem.

print("\nŠifrēts teksts:") # Informācija lietotājam par to, kā aizskatās šifrēts teksts (kuru mes atšifrēsim pēc atslēgas).

print(text\_sifrets) # Izvadam pagaidam nav atšifrētu tekstu, ka simbolu virkni.

print() # Atstarpe glītumam.

atslega = input\_atslega(text\_sifrets) # Prasam lietotājam ievādīt atslegu.

nav\_sifrets = decrypt(text\_sifrets, atslega, burti) # Atšifrējam nav\_sifrets simbolu virkni.

print()

print(f"Teksts tika atšifrēts ar atslēgu {atslega}:")

print(nav\_sifrets) # Izvadam atšifrētu tekstu lietotājam.

write\_text\_to\_file(nav\_sifrets\_txt, nav\_sifrets) # Ierakstam failā nav\_sifrets\_txt = "C:\\Users\\User\\Desktop\\nav\_sifrets\_2.txt" atšifrētu tekstu.

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

# Galvenā programmas daļa

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

nav\_sifrets\_txt = "C:\\Users\\User\\Desktop\\nav\_sifrets\_2.txt" # Šajā vietā lietotājs ievadīs tekstu kas nav šifrēts.

ir\_sifrets\_txt = "C:\\Users\\User\\Desktop\\ir\_sifrets\_2.txt" # Šajā vietā būs aizšifrēts teksts.

atslega\_txt = "C:\\Users\\User\\Desktop\\atslega\_2.txt" # Šajā vietā glabāsies atslega no aizšifrētam tekstam.

burti = ["A", "Ā", "B", "C", "Č", "D", "E", "Ē", "F", "G", "Ģ", "H", "I", "Ī", "J", "K", "Ķ", "L", "Ļ", "M", "N", "Ņ", "O", "P", "R", "S", "Š", "T", "U", "Ū", "V", "Z", "Ž", "\_"] # Latviešu alfabēts ar svitriņu.

# # ievade - boolean vertība, ja True, tad encryption (atšifrēts -> aizšifrēts). Ja False, tad decryption (aizšifrēts -> atšifrēts)

ievade = input\_cypher\_or\_decrypt() # Prasa lietotājam vai lietotājs grib aizšifrēt (encrypt) tekstu vai atšifrēt (decrypt) tekstu.

if ievade: # Ja ievade ir True, tad lietotājs grib aizšifrēt tekstu.

cypher\_main()

elif not ievade: # Ja ievade ir False, tad lietotājs grib atšifrēt tekstu.

decrypt\_main()

else:

print("Neparedzamā kļūda!")

**Testa piemēri:**

1)

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

2)

A computer screen with white text

Description automatically generated with low confidence

A picture containing text, screenshot, font

Description automatically generated

3)

A screen shot of a computer

Description automatically generated with medium confidence

A picture containing text, screenshot, font

Description automatically generated

**3. uzdevums**

Sastādīt programmu, kas veic teksta šifrēšanu un atšifrēšanu, izmantojot Morzes kodu.

Pieņemts, ka pārraidot signālu:

punkts - viena vienība, svītriņa - trīs vienības, pauze starp signāla vienumu - viena vienība pauze starp rakstzīmēm - trīs vienības, pauze starp vārdiem - septiņas vienības.

**Kods:**

# Programmas nosaukums: Morzes kods

# 3. uzdevums (1MPR15\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas veic teksta šifrēšanu un atšifrēšanu, izmantojot Morzes kodu.

# Pieņemts, ka pārraidot signālu:

# punkts - viena vienība, svītriņa - trīs vienības, pauze starp signāla vienumu - viena vienība

# pauze starp rakstzīmēm - trīs vienības, pauze starp vārdiem - septiņas vienības.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

def print\_text\_from\_data\_by\_rows(datne):

# Uzrakstā termināla lietotājam visu tekstu no .txt failā pa rindam.

# datne - datnes fails (piemēram, .txt fails)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

print(rinda, end='')

def save\_text\_from\_data\_by\_rows\_to\_variable(datne):

# Atgriež visu nolasītu tekstu no .txt datnes kā vienu str mainīgu.

# datne - datnes fails (piemēram, .txt fails).

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

a = ""

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

a = a + rinda

return a

def write\_text\_to\_file(filename, text):

# NODZES VISU INFORMĀCIJU filename DATNE un ieraksta jaunu informāciju no str text mainīga.

# text - str teksts, kuru gribam ierākstit datnē.

# filename - faila (datnes) nosaukums.

# Piemēram:

# filename = "C:\\Users\\User\\Desktop\\nav\_sifrets\_1.txt"

with open(filename, mode='w', encoding='utf-8') as file:

file.write(text)

def encrypt\_to\_morse\_code(non\_encrypted\_text, morse\_code\_dictionary):

# Funkcija, kas pārvērš neaizšifrētu tekstu par Morzes kodu, izmantojot Morzes kodu vārdnīcu.

# Atgriež aizšifrētu tekstu kā Morzes kodu (str).

# Ieraksta .txt datnē aizsifrets\_to\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods\_save.txt" tekstu, kas tika aizšifrēts.

# non\_encrypted\_text - str teksts, kuru gribam pārverst par Morzes kodu.

# morse\_code\_dictionary - vārdnica ar parastu simbolu : atbilstošu Morzes kodu

# Piemēram: {"A" : ".-", "B": "-..." , utt. }

encrypted\_text = "" # Sākumā aizšifrētais teksts ir tukšs.

for symbol in non\_encrypted\_text: # Pārskatām katru simbolu nešifrētajā tekstā.

sym = symbol.upper() # Pārveidojam simbolu tā, lai viņš būtu liels burts.

if sym in morse\_code\_dictionary: # Ja simbols ir Morzes koda vārdnīcā.

encrypted\_text = encrypted\_text + morse\_code\_dictionary[sym] + " " # Pievienojam aizšifrētu ar Morzes kodu burtu encrypted\_text simbolu virknei ar atstarpi.

else:

encrypted\_text = encrypted\_text + sym + " " # Ja simbols nav Morzes kodā, tad vienkarši pievienojam to nešifrētu burtu kopā ar atstarpi.

write\_text\_to\_file(aizsifrets\_to\_morses\_kods\_save\_txt, encrypted\_text) # Ieraksta .txt datnē aizsifrets\_to\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods\_save.txt" tekstu, kas tika atšifrēts.

return encrypted\_text # Atgriežam aizšifrētu Morzes kodu.

def decrypt\_from\_morse\_code(encrypted\_text, morse\_code\_dictionary):

# Funkcija, kas atšifrē Morzes kodu (pārverš par vienkaršu str tekstu), izmantojot Morzes kodu vārdnīcu.

# Atgriež atšifrētu tekstu no Morzes kodu kā parastu tekstu (str).

# Ieraksta .txt datnē atsifrets\_no\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\atsifrets\_no\_morzes\_koda\_save.txt" tekstu, kas tika atšifrēts.

# encrypted\_text - str Morzes kods, kuru gribam pārverst par parastu tekstu.

# morse\_code\_dictionary - vārdnica ar parastu simbolu : atbilstošu Morzes kodu

# Piemēram: {"A" : ".-", "B": "-..." , utt. }

decrypted\_text = "" # Sākumā atšifrētais teksts ir tukšs.

morse\_code = "" # Sākumā Morzes kods ir tukšs.

for symbol in encrypted\_text: # Pārskatām katru simbolu aizšifrētajā tekstā.

if symbol != " ": # Ja simbols nav atstarpe.

morse\_code = morse\_code + symbol # Pievienojam simbolu Morzes kodam.

else:

if morse\_code in morse\_code\_dictionary.values(): # Ja Morzes kods ir atrodams Morzes koda vārdnīcā.

for key, value in morse\_code\_dictionary.items(): # Pārskatām katru pāri (atslēga, vērtība) vārdnīcā.

if value == morse\_code: # Ja vērtība atbilst Morzes kodam.

decrypted\_text = decrypted\_text + key # Pievienojam vārdnicas noteiktu "atslēgu" (key) (key - value vārdnīca) atšifrētajam tekstam.

break # Pārtraucam meklēšanu pēc "atslēgas" (key).

else:

decrypted\_text = decrypted\_text + morse\_code # Ja Morzes kods nav atrodams vārdnīcā, pievienojam to atšifrētajam tekstam.

morse\_code = ""

# Pārbaudam, vai teksta non\_encrypted\_text beigās nav palicis Morzes kods.

if morse\_code != "":

if morse\_code in morse\_code\_dictionary.values(): # Ja Morzes kods ir atrodams vārdnīcā.

for key, value in morse\_code\_dictionary.items(): # Pārskatām katru pāri (atslēga, vērtība) vārdnīcā.

if value == morse\_code: # Ja vērtība atbilst Morzes kodam.

decrypted\_text = decrypted\_text + key # Pievienojam vārdnicas noteiktu "atslēgu" (key) (key - value vārdnīca) atšifrētajam tekstam.

break # Pārtraucam meklēšanu pēc "atslēgas" (key).

else:

decrypted\_text = decrypted\_text + morse\_code # Ja Morzes kods nav atrodams vārdnīcā, pievienojam to atšifrētajam tekstam.

write\_text\_to\_file(atsifrets\_no\_morses\_kods\_save\_txt, decrypted\_text) # Ieraksta .txt datnē atsifrets\_no\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\atsifrets\_no\_morzes\_koda\_save.txt" tekstu, kas tika atšifrēts.

return decrypted\_text # Atgriežam atšifrēto tekstu

def input\_cypher\_to\_morse\_or\_decrypt\_from\_morse():

# Prasa lietotājam vai lietotājs grib aizšifrēt (encrypt) tekstu vai atšifrēt (decrypt) tekstu.

# Ja lietotājs ievādīs "c" vai "C", tad viņš grib aizšifrēt (encrypt).

# Ja lietotājs ievādīs "d" vai "D", tad viņš grib atšifrēt (decrypt).

# Atgriež True, ja ir ievādīts "c" vai "C" (str).

# Atgriež False, ja ir ievādīts "d" vai "D" (str).

cypher\_or\_decrypt = ""

while cypher\_or\_decrypt.lower() != "c" and cypher\_or\_decrypt.lower() != "d":

cypher\_or\_decrypt = input("Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> ")

if cypher\_or\_decrypt.lower() == "c":

return True

elif cypher\_or\_decrypt.lower() == "d":

return False

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

# Galvenā programmas daļa

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

# Morzes koda avots:

# https://en.wikipedia.org/wiki/Morse\_code#/media/File:International\_Morse\_Code.svg

# Tiek izmantota atstārpes rakstzīme " ", lai paradītu pauzi starp rakstzīmēm - trīs vienības.

# Tiek izmantota atstārpes rakstzīme "/", lai paradītu pauzi starp vārdiem - septiņas vienības.

# "." - viena vienība.

# " " - trīs vienības.

# "/" - septiņas vienības, lai paradītu atstārpi " " starp vārdiem, atstājam to ar "/".

morse\_code\_dictionary = {

"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": ".", "F": "..-.", "G": "--.", "H": "....",

"I": "..", "J": ".---", "K": "-.-", "L": ".-..", "M": "--", "N": "-.", "O": "---", "P": ".--.",

"Q": "--.-", "R": ".-.", "S": "...", "T": "-", "U": "..-", "V": "...-", "W": ".--", "X": "-..-",

"Y": "-.--", "Z": "--..",

"1": ".----", "2": "..---", "3": "...--", "4": "....-", "5": ".....", '6': "-....", "7": "--...", "8": "---..", "9": "----.", "0": "-----",

" ": "/" # 3 vienības.

}

nav\_sifrets\_to\_morse\_txt = "C:\\Users\\User\\Desktop\\teksts\_to\_morze.txt" # Šajā vietā lietotājs ievadīs tekstu kas nav šifrēts (ievada lietotājs).

ir\_sifrets\_to\_morse\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods.txt" # Šajā vietā būs aizšifrēts teksts ar Morzes kodu (ievada lietotājs).

aizsifrets\_to\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods\_save.txt" # Šajā vietā būs aizšifrēts teksts ar Morzes kodu (programma šajā datnē ieraksta aizšifrētu tekstu ar Morzes kodu).

atsifrets\_no\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\atsifrets\_no\_morzes\_koda\_save.txt" # Šajā vietā būs atšifrēts teksts no Morzes koda (programma šajā datnē ieraksta atšifrētu tekstu no Morzes koda).

print("Teksts kurš ir ierakstīts teksts\_to\_morze.txt datnē un kuru var aizšifrēt:")

print\_text\_from\_data\_by\_rows(nav\_sifrets\_to\_morse\_txt)

print("\n\nTeksts kurš ir ierakstīts morzes\_kods.txt datnē un kuru var atšifrēt:")

print\_text\_from\_data\_by\_rows(ir\_sifrets\_to\_morse\_txt)

print("\n\nJa gribat atšifrēt vai aizšifrēt citu tekstu, tad izmainiet atbilstošu datnes saturu.\n")

print("---------------------------------------------------------------------------------------\n")

ievade = input\_cypher\_to\_morse\_or\_decrypt\_from\_morse()

if ievade: # Ja ievade ir True, tad lietotājs grib aizšifrēt tekstu.

message\_to\_encrypt = save\_text\_from\_data\_by\_rows\_to\_variable(nav\_sifrets\_to\_morse\_txt) # Saglabājam teksts\_to\_morze.txt datnes saturu kā str mainīgu message\_to\_encrypt

encrypted\_text = encrypt\_to\_morse\_code(message\_to\_encrypt, morse\_code\_dictionary) # Aizšifrējam str tekstu message\_to\_encrypt

print("\nTeksts:")

print\_text\_from\_data\_by\_rows(nav\_sifrets\_to\_morse\_txt)

print("\n\nAizšifrēts teksts ar Morzes kodu:")

print(encrypted\_text) # Izvadīt aizšifrētu ar Morzes kodu tekstu.

elif not ievade: # Ja ievade ir False, tad lietotājs grib atšifrēt tekstu.

message\_to\_decrypt = save\_text\_from\_data\_by\_rows\_to\_variable(ir\_sifrets\_to\_morse\_txt) # Saglabājam morzes\_kods.txt datnes saturu kā str mainīgu message\_to\_decrypt

decrypted\_message = decrypt\_from\_morse\_code(message\_to\_decrypt, morse\_code\_dictionary) # Atšifrējam str tekstu message\_to\_decrypt.

print("\nAizšifrēts teksts ar Morzes kodu:")

print\_text\_from\_data\_by\_rows(ir\_sifrets\_to\_morse\_txt)

print("\n\nAtšifrēts teksts:")

print(decrypted\_message) # Izvadīt atšifrētu tekstu no Morzes koda.

else:

print("Neparedzamā kļūda!")

**Testa piemēri:**

1)

A picture containing text, screenshot, font, software

Description automatically generated

2)

A picture containing text, screenshot, font

Description automatically generated

3)

A picture containing text, screenshot, font

Description automatically generated

**PU3. uzdevums**

Sastādīt programmu, kas veic teksta šifrēšanu, izmantojot Mores kodu, kodu pārraidot kā skaņas un/vai gaismas signālu.

**Kods:**

# Programmas nosaukums: Morzes kods ar pīksteņiem

# PU3. uzdevums (1MPR15\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas veic teksta šifrēšanu, izmantojot Mores kodu, kodu pārraidot kā skaņas un/vai gaismas signālu.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import winsound

def play\_beep(duration):

# Atskaņo pīksteņu ar ilgumu duration.

# duration - pīksteņu ilgums milisekundes (int).

frequency = 700 # Pīkstienu frekvence Hz

winsound.Beep(frequency, duration)

def morse\_play(message):

# Atskaņo Morzes kodu ar pīksteņiem.

# message - Morzes koda str virkne.

for symbol in message: # Iet cikla pa katru simbolu in message. Ja nav zināms simbols, tad atskaņo neko.

if symbol == ".":

play\_beep(dot\_duration)

elif symbol == "-":

play\_beep(dash\_duration)

elif symbol == " ":

play\_beep(pause\_between\_letters\_duration)

elif symbol == "/":

play\_beep(pause\_between\_words\_duration)

def print\_text\_from\_data\_by\_rows(datne):

# Uzrakstā termināla lietotājam visu tekstu no .txt failā pa rindam.

# datne - datnes fails (piemēram, .txt fails)

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

print(rinda, end='')

def save\_text\_from\_data\_by\_rows\_to\_variable(datne):

# Atgriež visu nolasītu tekstu no .txt datnes kā vienu str mainīgu.

# datne - datnes fails (piemēram, .txt fails).

# Piemēram:

# datne = "C:\\Users\\User\\Desktop\\teksts.txt"

a = ""

with open(datne, mode="r", encoding="utf-8") as datne:

for rinda in datne:

a = a + rinda

return a

def write\_text\_to\_file(filename, text):

# NODZES VISU INFORMĀCIJU filename DATNE un ieraksta jaunu informāciju no str text mainīga.

# text - str teksts, kuru gribam ierākstit datnē.

# filename - faila (datnes) nosaukums.

# Piemēram:

# filename = "C:\\Users\\User\\Desktop\\nav\_sifrets\_1.txt"

with open(filename, mode='w', encoding='utf-8') as file:

file.write(text)

def encrypt\_to\_morse\_code(non\_encrypted\_text, morse\_code\_dictionary):

# Funkcija, kas pārvērš neaizšifrētu tekstu par Morzes kodu, izmantojot Morzes kodu vārdnīcu.

# Atgriež aizšifrētu tekstu kā Morzes kodu (str).

# Ieraksta .txt datnē aizsifrets\_to\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods\_save.txt" tekstu, kas tika aizšifrēts.

# non\_encrypted\_text - str teksts, kuru gribam pārverst par Morzes kodu.

# morse\_code\_dictionary - vārdnica ar parastu simbolu : atbilstošu Morzes kodu

# Piemēram: {"A" : ".-", "B": "-..." , utt. }

encrypted\_text = "" # Sākumā aizšifrētais teksts ir tukšs.

for symbol in non\_encrypted\_text: # Pārskatām katru simbolu nešifrētajā tekstā.

sym = symbol.upper() # Pārveidojam simbolu tā, lai viņš būtu liels burts.

if sym in morse\_code\_dictionary: # Ja simbols ir Morzes koda vārdnīcā.

encrypted\_text = encrypted\_text + morse\_code\_dictionary[sym] + " " # Pievienojam aizšifrētu ar Morzes kodu burtu encrypted\_text simbolu virknei ar atstarpi.

else:

encrypted\_text = encrypted\_text + sym + " " # Ja simbols nav Morzes kodā, tad vienkarši pievienojam to nešifrētu burtu kopā ar atstarpi.

write\_text\_to\_file(aizsifrets\_to\_morses\_kods\_save\_txt, encrypted\_text) # Ieraksta .txt datnē aizsifrets\_to\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods\_save.txt" tekstu, kas tika atšifrēts.

return encrypted\_text # Atgriežam aizšifrētu Morzes kodu.

def decrypt\_from\_morse\_code(encrypted\_text, morse\_code\_dictionary):

# Funkcija, kas atšifrē Morzes kodu (pārverš par vienkaršu str tekstu), izmantojot Morzes kodu vārdnīcu.

# Atgriež atšifrētu tekstu no Morzes kodu kā parastu tekstu (str).

# Ieraksta .txt datnē atsifrets\_no\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\atsifrets\_no\_morzes\_koda\_save.txt" tekstu, kas tika atšifrēts.

# encrypted\_text - str Morzes kods, kuru gribam pārverst par parastu tekstu.

# morse\_code\_dictionary - vārdnica ar parastu simbolu : atbilstošu Morzes kodu

# Piemēram: {"A" : ".-", "B": "-..." , utt. }

decrypted\_text = "" # Sākumā atšifrētais teksts ir tukšs.

morse\_code = "" # Sākumā Morzes kods ir tukšs.

for symbol in encrypted\_text: # Pārskatām katru simbolu aizšifrētajā tekstā.

if symbol != " ": # Ja simbols nav atstarpe.

morse\_code = morse\_code + symbol # Pievienojam simbolu Morzes kodam.

else:

if morse\_code in morse\_code\_dictionary.values(): # Ja Morzes kods ir atrodams Morzes koda vārdnīcā.

for key, value in morse\_code\_dictionary.items(): # Pārskatām katru pāri (atslēga, vērtība) vārdnīcā.

if value == morse\_code: # Ja vērtība atbilst Morzes kodam.

decrypted\_text = decrypted\_text + key # Pievienojam vārdnicas noteiktu "atslēgu" (key) (key - value vārdnīca) atšifrētajam tekstam.

break # Pārtraucam meklēšanu pēc "atslēgas" (key).

else:

decrypted\_text = decrypted\_text + morse\_code # Ja Morzes kods nav atrodams vārdnīcā, pievienojam to atšifrētajam tekstam.

morse\_code = "" # Morzes kods atkal ir tukšs.

# Pārbaudam, vai teksta non\_encrypted\_text beigās nav palicis Morzes kods.

if morse\_code != "":

if morse\_code in morse\_code\_dictionary.values(): # Ja Morzes kods ir atrodams vārdnīcā.

for key, value in morse\_code\_dictionary.items(): # Pārskatām katru pāri (atslēga, vērtība) vārdnīcā.

if value == morse\_code: # Ja vērtība atbilst Morzes kodam.

decrypted\_text = decrypted\_text + key # Pievienojam vārdnicas noteiktu "atslēgu" (key) (key - value vārdnīca) atšifrētajam tekstam.

break # Pārtraucam meklēšanu pēc "atslēgas" (key).

else:

decrypted\_text = decrypted\_text + morse\_code # Ja Morzes kods nav atrodams vārdnīcā, pievienojam to atšifrētajam tekstam.

write\_text\_to\_file(atsifrets\_no\_morses\_kods\_save\_txt, decrypted\_text) # Ieraksta .txt datnē atsifrets\_no\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\atsifrets\_no\_morzes\_koda\_save.txt" tekstu, kas tika atšifrēts.

return decrypted\_text # Atgriežam atšifrēto tekstu

def input\_cypher\_to\_morse\_or\_decrypt\_from\_morse():

# Prasa lietotājam vai lietotājs grib aizšifrēt (encrypt) tekstu vai atšifrēt (decrypt) tekstu.

# Ja lietotājs ievādīs "c" vai "C", tad viņš grib aizšifrēt (encrypt).

# Ja lietotājs ievādīs "d" vai "D", tad viņš grib atšifrēt (decrypt).

# Atgriež True, ja ir ievādīts "c" vai "C" (str).

# Atgriež False, ja ir ievādīts "d" vai "D" (str).

cypher\_or\_decrypt = ""

while cypher\_or\_decrypt.lower() != "c" and cypher\_or\_decrypt.lower() != "d":

cypher\_or\_decrypt = input("Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> ")

if cypher\_or\_decrypt.lower() == "c":

return True

elif cypher\_or\_decrypt.lower() == "d":

return False

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

# Galvenā programmas daļa

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

# Morzes koda avots:

# https://en.wikipedia.org/wiki/Morse\_code#/media/File:International\_Morse\_Code.svg

# Tiek izmantota atstārpes rakstzīme " ", lai paradītu pauzi starp rakstzīmēm - trīs vienības.

# Tiek izmantota atstārpes rakstzīme "/", lai paradītu pauzi starp vārdiem - septiņas vienības.

# "." - viena vienība.

# " " - trīs vienības.

# "/" - septiņas vienības, lai paradītu atstārpi " " starp vārdiem, atstājam to ar "/".

morse\_code\_dictionary = {

"A": ".-", "B": "-...", "C": "-.-.", "D": "-..", "E": ".", "F": "..-.", "G": "--.", "H": "....",

"I": "..", "J": ".---", "K": "-.-", "L": ".-..", "M": "--", "N": "-.", "O": "---", "P": ".--.",

"Q": "--.-", "R": ".-.", "S": "...", "T": "-", "U": "..-", "V": "...-", "W": ".--", "X": "-..-",

"Y": "-.--", "Z": "--..",

"1": ".----", "2": "..---", "3": "...--", "4": "....-", "5": ".....", '6': "-....", "7": "--...", "8": "---..", "9": "----.", "0": "-----",

" ": "/" # 3 vienības.

}

# Morzes koda ilgums milisekundēs

dot\_duration = 120 # Punkts - viena vienība.

dash\_duration = dot\_duration \* 3 # Svītriņa - trīs vienības.

pause\_between\_letters\_duration = dot\_duration # Pauze starp signāla vienumu - viena vienība.

pause\_between\_words\_duration = dot\_duration \* 7 # Pauze starp vārdiem - septiņas vienības.

nav\_sifrets\_to\_morse\_txt = "C:\\Users\\User\\Desktop\\teksts\_to\_morze.txt" # Šajā vietā lietotājs ievadīs tekstu kas nav šifrēts (ievada lietotājs).

ir\_sifrets\_to\_morse\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods.txt" # Šajā vietā būs aizšifrēts teksts ar Morzes kodu (ievada lietotājs).

aizsifrets\_to\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\morzes\_kods\_save.txt" # Šajā vietā būs aizšifrēts teksts ar Morzes kodu (programma šajā datnē ieraksta aizšifrētu tekstu ar Morzes kodu).

atsifrets\_no\_morses\_kods\_save\_txt = "C:\\Users\\User\\Desktop\\atsifrets\_no\_morzes\_koda\_save.txt" # Šajā vietā būs atšifrēts teksts no Morzes koda (programma šajā datnē ieraksta atšifrētu tekstu no Morzes koda).

print("Teksts kurš ir ierakstīts teksts\_to\_morze.txt datnē un kuru var aizšifrēt:")

print\_text\_from\_data\_by\_rows(nav\_sifrets\_to\_morse\_txt)

print("\n\nTeksts kurš ir ierakstīts morzes\_kods.txt datnē un kuru var atšifrēt:")

print\_text\_from\_data\_by\_rows(ir\_sifrets\_to\_morse\_txt)

print("\n\nJa gribat atšifrēt vai aizšifrēt citu tekstu, tad izmainiet atbilstošu datnes saturu.\n")

print("---------------------------------------------------------------------------------------\n")

ievade = input\_cypher\_to\_morse\_or\_decrypt\_from\_morse()

if ievade: # Ja ievade ir True, tad lietotājs grib aizšifrēt tekstu.

message\_to\_encrypt = save\_text\_from\_data\_by\_rows\_to\_variable(nav\_sifrets\_to\_morse\_txt) # Saglabājam teksts\_to\_morze.txt datnes saturu kā str mainīgu message\_to\_encrypt

encrypted\_text = encrypt\_to\_morse\_code(message\_to\_encrypt, morse\_code\_dictionary) # Aizšifrējam str tekstu message\_to\_encrypt

print("\nTeksts:")

print\_text\_from\_data\_by\_rows(nav\_sifrets\_to\_morse\_txt)

print("\n\nAizšifrēts teksts ar Morzes kodu:")

print(encrypted\_text) # Izvadīt aizšifrētu ar Morzes kodu tekstu.

morse\_play(encrypted\_text)

elif not ievade: # Ja ievade ir False, tad lietotājs grib atšifrēt tekstu.

message\_to\_decrypt = save\_text\_from\_data\_by\_rows\_to\_variable(ir\_sifrets\_to\_morse\_txt) # Saglabājam morzes\_kods.txt datnes saturu kā str mainīgu message\_to\_decrypt

decrypted\_message = decrypt\_from\_morse\_code(message\_to\_decrypt, morse\_code\_dictionary) # Atšifrējam str tekstu message\_to\_decrypt.

print("\nAizšifrēts teksts ar Morzes kodu:")

print\_text\_from\_data\_by\_rows(ir\_sifrets\_to\_morse\_txt)

print("\n\nAtšifrēts teksts:")

print(decrypted\_message) # Izvadīt atšifrētu tekstu no Morzes koda.

morse\_play(message\_to\_decrypt)

else:

print("Neparedzamā kļūda!")

**Testa piemēri:**

1)

Video ar pikstēšanu:

A picture containing text, screenshot, font

Description automatically generated<https://youtu.be/HbYfGPmI2Jg>

2)

Video ar pikstēšanu:

A screen shot of a computer

Description automatically generated with low confidence<https://youtu.be/YDbcTupV02E>

3)

Video ar pikstēšanu:

A picture containing screenshot, text, black

Description automatically generated<https://youtu.be/qfoF6TskBJY>