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:

1)

```
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
```

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

2) Tekstu avoti ir pieminēti .txt failā.
.txt fails:
liels_teksts_eng.txt


liels_teksts_eng.txt - Notepad
File Edit Format View Help
By Derek Cai & Annabelle Liang
BBC News
Artificial intelligence has emerged as enough of a concern that it made it onto what was already a packed agenda at the G7 summit at the weekend.

Concerns about AI's harmful impact coincide with the US' attempts to restrict China's access to crucial technology.

For now, the US seems to be ahead in the AI race. And there is already the possibility that current restrictions on semiconductor exports to China could hamper Beijing's technological progress.

But China could catch up, according to analysts, as AI solutions take years to be perfected. Chinese internet companies "are arguably more advanced than US internet companies, depending on how you're measuring advancement," Kendra Schaef

However, she says China's "ability to manufacture high-end equipment and components is an estimated 10 to 15 years behind global leaders."

The Silicon Valley factor
The US' biggest advantage is Silicon Valley, arguably the world's supreme entrepreneurial hotspot. It is the birthplace of technology giants such as Google, Apple and Intel that have helped shape modern life.

Innovators in the country have been helped by its unique research culture, says Pascale Fung, director of the Center for Artificial Intelligence Research at the Hong Kong University of Science and Technology.

Researchers often spend years working to improve a technology without a product in mind, Ms Fung says.

OpenAI, for example, operated as a non-profit company for years as it researched the Transformers machine learning model, which eventually powered ChatGPT.

"This environment never existed in most Chinese companies. They would build deep learning systems or large language models only after they saw the popularity," she adds. "This is a fundamental challenge to Chinese AI."

US investors have also been supportive of the country's research push. In 2019, Microsoft said it would put $1bn (£810,000) in to OpenAI.

"AI is one of the most transformative technologies of our time and has the potential to help solve many of our world's most pressing challenges," Microsoft chief executive Satya Nadella said.

China's edge
China, meanwhile, benefits from a larger consumer base. It is the world's second-most populous country, home to roughly 1.4 billion people.

It also has a thriving internet sector, says Edith Yeung, a partner at the Race Capital investment firm.

Nearly everyone in the country uses the super app WeChat, for example. It is used for almost everything from sending text messages, to booking doctor's appointments and filing taxes.

As a result, there's a wealth of information that can be used to improve products. "The AI model is going to be only as good as the data that is available for it to learn from," Ms Yeung says.

"For good or bad, China has a lot less rules around privacy, and a lot more data [compared to the US]. There's CCTV facial recognition everywhere, for example," she adds. "Imagine how useful that would be for AI-generated images."

While China's tech community may appear to be lagging behind the US, its developers have an edge, according to Lee Kai-Fu, who makes the argument in his book AI Superpowers: China, Silicon Valley, and the New World Order.

"They live in a world where speed is essential, copying is an accepted practice, and competitors will stop at nothing to win a new market," wrote Mr Lee, a prominent figure in Beijing's internet sector and the former head of Google China

"This rough-and-tumble environment makes a strong contrast to Silicon Valley, where copying is stigmatised and many companies are allowed to coast on the basis of one original idea or lucky break."

China's copycat era has its problems, including serious issues around intellectual property. Mr Lee writes that it has led to a generation of hardy and nimble entrepreneurs ready to compete.

Since the 1980s, China has been expanding its economy, which used to be based mainly on manufacturing, to one that is technology-based, Ms Fung says.

"In the last decade, we have seen more innovation from Chinese consumer-driven internet companies and high-end Chinese designs," she adds.

Can China catch up?
While Chinese tech companies certainly have unique advantages, the full impact of Beijing's authoritarianism is still unclear.

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

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 terminmāla 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)

```
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
```

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

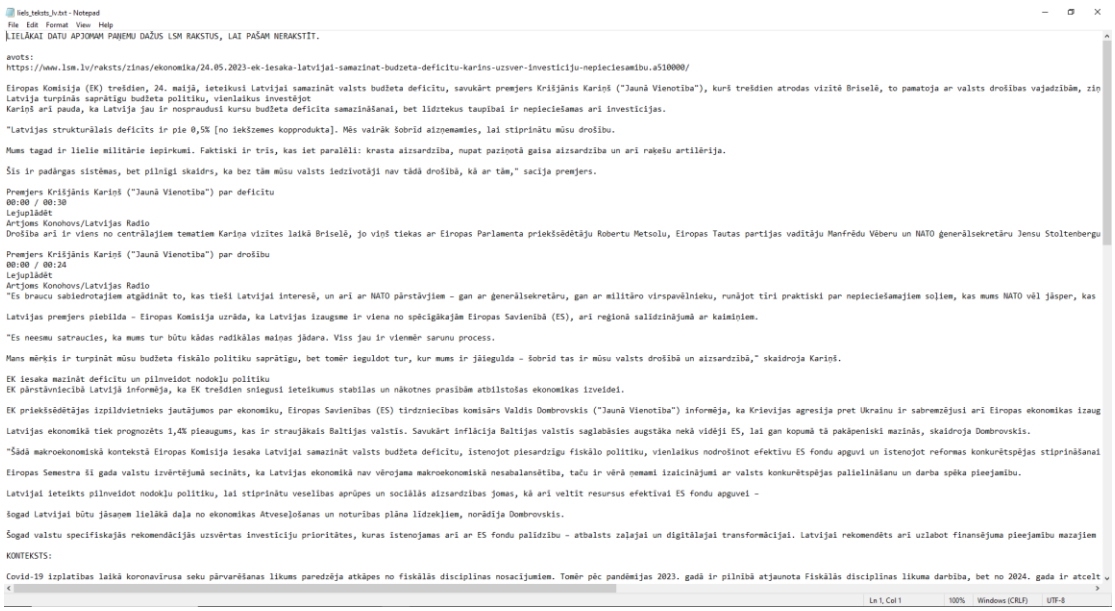
Č 11

Ķ 7

3) Tekstu avoti ir pieminēti .txt failā.

.txt fails:

liels_teksts_lv.txt


liels_teksts_lv.txt - Notepad

File Edit Format View Help

LIELĀKAI DATU APJOMAM PAŅEMU DAŽUS LSM RAKSTUS, LAI PAŠAM NERAKSTĪT.

avots:
https://www.lsm.lv/raksts/zinas/ekonomika/24.05.2023-ek-iesaka-latvijai-samazinat-budzeta-deficitu-karins-uzsver-investiciju-nepieciesamibu.a510000/

Eiropas Komisija (EK) trešdien, 24. maijā, ieteikusi Latvijai samazināt valsts budžeta deficītu, savukārt premjers Krišjānis Kariņš ("Jaunā Vienotība"), kurš trešdien atrodas vizītē Briselē, to pamatoja ar valsts drošības vajadzībām, zi
Latvija turpinās saprātīgu budžeta politiku, vienlaikus investējot
Kariņš arī pauda, ka Latvija jau ir nospraususi kursu budžeta deficīta samazināšanai, bet līdztekus taupībai ir nepieciešamas arī investīcijas.

"Latvijas strukturālais deficīts ir pie 0,5% [no iekšzemes kopprodukta]. Mēs vairāk šobrīd aizņemamies, lai stiprinātu mūsu drošību.

Mums tagad ir lielie militārie iepirkumi. Faktiski ir trīs, kas iet paralēli: krasta aizsardzība, nupat paziņotā gaisa aizsardzība un arī raķešu artilērija.

Šīs ir padārgas sistēmas, bet pilnīgi skaidrs, ka bez tām mūsu valsts iedzīvotāji nav tādā drošībā, kā ar tām," sacīja premjers.

Premjers Krišjānis Kariņš ("Jaunā Vienotība") par deficītu
00:00 / 00:30
Lejuplādēt
Artjoms Konohovs/Latvijas Radio
Drošība arī ir viens no centrālajiem tematiem Kariņa vizītes laikā Briselē, jo viņš tiekas ar Eiropas Parlamenta priekšsēdētāju Robertu Metsolu, Eiropas Tautas partijas vadītāju Manfrēdu Vēberu un NATO ģenerālsekretāru Jensu Stoltenbergu

Premjers Krišjānis Kariņš ("Jaunā Vienotība") par drošību
00:00 / 00:24
Lejuplādēt
Artjoms Konohovs/Latvijas Radio
"Es braucu sabiedrotajiem atgādināt to, kas tieši Latvijai interesē, un arī ar NATO pārstāvjiem – gan ar ģenerālsekretāru, gan ar militāro virspavēlnieku, runājot tiri praktiski par nepieciešamajiem soļiem, kas mums NATO vēl jāspēr, kas

Latvijas premjers piebilda – Eiropas Komisija uzrāda, ka Latvijas izaugsme ir viena no spēcīgākajām Eiropas Savienībā (ES), arī reģionā salīdzinājumā ar kaimiņiem.

"Es neesmu satraucies, ka mums tur būtu kādas radikālas maiņas jādara. Viss jau ir vienmēr sarunu process.

Mans mērķis ir turpināt mūsu budžeta fiskālo politiku saprātīgu, bet tomēr ieguldot tur, kur mums ir jāiegulda – šobrīd tas ir mūsu valsts drošībā un aizsardzībā," skaidroja Kariņš.

EK iesaka mazināt deficītu un pilnveidot nodokļu politiku
EK pārstāvniecībā Latvijā informēja, ka EK trešdien sniegusi ieteikumus stabilas un nākotnes prasībām atbilstošas ekonomikas izveidei.

EK priekšsēdētājas izpildvietnieks jautājumos par ekonomiku, Eiropas Savienības (ES) tirdzniecības komisārs Valdis Dombrovskis ("Jaunā Vienotība") informēja, ka Krievijas agresija pret Ukrainu ir sabremzējusi arī Eiropas ekonomikas izaug

Latvijas ekonomikā tiek prognozēts 1,4% pieaugums, kas ir straujākais Baltijas valstīs. Savukārt inflācija Baltijas valstīs saglabāsies augstāka nekā vidēji ES, lai gan kopumā tā pakāpeniski mazinās, skaidroja Dombrovskis.

"Šādā makroekonomiskā kontekstā Eiropas Komisija iesaka Latvijai samazināt valsts budžeta deficītu, īstenojot piesardzīgu fiskālo politiku, vienlaikus nodrošinot efektīvu ES fondu apguvi un īstenojot reformas konkurētspējas stiprināšanai

Eiropas Semestra šī gada valstu izvērtējumā secināts, ka Latvijas ekonomikā nav vērojama makroekonomiskā nesabalansētība, taču ir vērā ņemami izaicinājumi ar valsts konkurētspējas palielināšanu un darba spēka pieejamību.

Latvijai ieteikts pilnveidot nodokļu politiku, lai stiprinātu veselības aprūpes un sociālās aizsardzības jomas, kā arī veltīt resursus efektīvai ES fondu apguvei –

šogad Latvijai būtu jāsaņem lielākā daļa no ekonomikas Atveseļošanas un noturības plāna līdzekļiem, norādīja Dombrovskis.

Šogad valstu specifiskajās rekomendācijās uzsvērtas investīciju prioritātes, kuras īstenojamas arī ar ES fondu palīdzību – atbalsts zaļajai un digitālajai transformācijai. Latvijai rekomendēts arī uzlabot finansējuma pieejamību mazajiem

KONTEKSTS:

Covid-19 izplatības laikā koronavīrusa seku pārvarēšanas likums paredzēja atkāpes no fiskālās disciplīnas nosacījumiem. Tomēr pēc pandēmijas 2023. gadā ir pilnībā atjaunota Fiskālās disciplīnas likuma darbība, bet no 2024. gada ir atcelt

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

2. uzdevums

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

JĀNIS_IR_TEICAMNIEKS

19651965196519651965

KĢŠLŠFĻUABILČGSSĪĶŅV

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

```
# KĢŠLŠFĻUABILČGSSĪĶŅV
# 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)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> c

Nav šifrēts teksts:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET

Ievadiet atslēgu ==> 1

Teksts tika aizšifrēts ar atslēgu 1:
MBCEJFO, TJ JS TMFQFOB _JOB! MPUJ TWBSJHJ, MPUJ TWBSJHJ, MPUJ TMFQFOJ! 25/05/2999
UPQ TFDSFU
UPQ TFDSFU
UPQ TFDSFU
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
MBCEJFO, TJ JS TMFQFOB _JOB! MPUJ TWBSJHJ, MPUJ TWBSJHJ, MPUJ TMFQFOJ! 25/05/2999
UPQ TFDSFU
UPQ TFDSFU
UPQ TFDSFU

Ievadiet atslēgu ==> 1

Teksts tika atšifrēts ar atslēgu 1:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET
```

2)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> asf
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> asfa
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> labi
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> C

Nav šifrēts teksts:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET

Ievadiet atslēgu ==> 0

Teksts tika aizšifrēts ar atslēgu 0:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET

Ievadiet atslēgu ==> asf
Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!
Ievadiet atslēgu ==> -1
Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!
Ievadiet atslēgu ==> -111
Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!
Ievadiet atslēgu ==> 0

Teksts tika atšifrēts ar atslēgu 0:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET
```

3)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> gig
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> c

Nav šifrēts teksts:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET

Ievadiet atslēgu ==> -1
Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!
Ievadiet atslēgu ==> 123456789101112

Teksts tika aizšifrēts ar atslēgu 123456789101112:
MCEHNKU, TI JS TNHTJTH HJNB! NPVL XAHZRHI, MQUK W_GYQPJ, MPVJ VPJVLVR! 25/05/2999
AXQ TFDTFV
XTV _NDRFU
VPR WJIYMB
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
MCEHNKU, TI JS TNHTJTH HJNB! NPVL XAHZRHI, MQUK W_GYQPJ, MPVJ VPJVLVR! 25/05/2999
AXQ TFDTFV
XTV _NDRFU
VPR WJIYMB

Ievadiet atslēgu ==> 123456789101112

Teksts tika atšifrēts ar atslēgu 123456789101112:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET
```

4)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> c

Nav šifrēts teksts:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET

Ievadiet atslēgu ==> 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!
Ievadiet atslēgu ==> 123

Teksts tika aizšifrēts ar atslēgu 123:
MCEEKHO, TK JT TNHQGQB BJPD! OPVL UYBTLHK, NRUK TXDSKJJ, MQWJ VMGSFPL! 25/05/2999
VRQ VFEUFV
UQS UHDTHU
WPR TGFSGW
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
MCEEKHO, TK JT TNHQGQB BJPD! OPVL UYBTLHK, NRUK TXDSKJJ, MQWJ VMGSFPL! 25/05/2999
VRQ VFEUFV
UQS UHDTHU
WPR TGFSGW

Ievadiet atslēgu ==> 123

Teksts tika atšifrēts ar atslēgu 123:
LABDIEN, SI IR SLEPENA ZINA! LOTI SVARIGI, LOTI SVARIGI, LOTI SLEPENI! 25/05/2999
TOP SECRET
TOP SECRET
TOP SECRET
```

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)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> c

Nav šifrēts teksts:
LABDIEN! SLEPENA INFORMĀCIJA! ĻOTI SVARĪGI, ĻOTI SVARĪGI, PILNĪGI SLEPENI! 25/05/2999
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS

AĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ

Ievadiet atslēgu ==> 1

Teksts tika aizšifrēts ar atslēgu 1:
ĻĀCEĪĒŅ! ŠĻĒRĒŅĀ ĪŅGPSNBČĪKĀ! MPUĪ ŠZĀSJĢĪ, MPUĪ ŠZĀSJĢĪ, RĪĻŅJĢĪ ŠĻĒRĒŅĪ! 25/05/2999
ĻĪĒĻŠ ŅPŠĻFRŪNŠ
ĻĪĒĻŠ ŅPŠĻFRŪNŠ
ĻĪĒĻŠ ŅPŠĻFRŪNŠ

ĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ_
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
ĻĀCEĪĒŅ! ŠĻĒRĒŅĀ ĪŅGPSNBČĪKĀ! MPUĪ ŠZĀSJĢĪ, MPUĪ ŠZĀSJĢĪ, RĪĻŅJĢĪ ŠĻĒRĒŅĪ! 25/05/2999
ĻĪĒĻŠ ŅPŠĻFRŪNŠ
ĻĪĒĻŠ ŅPŠĻFRŪNŠ
ĻĪĒĻŠ ŅPŠĻFRŪNŠ

ĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ_

Ievadiet atslēgu ==> 1

Teksts tika atšifrēts ar atslēgu 1:
LABDIEN! SLEPENA INFORMĀCIJA! ĻOTI SVARĪGI, ĻOTI SVARĪGI, PILNĪGI SLEPENI! 25/05/2999
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS

AĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ
```

2)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> c

Nav šifrēts teksts:
LABDIEN! SLEPENA INFORMĀCIJA! ĻOTI SVARĪGI, ĻOTI SVARĪGI, PILNĪGI SLEPENI! 25/05/2999
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS

AĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ

Ievadiet atslēgu ==> 123456789

Teksts tika aizšifrēts ar atslēgu 123456789:
ĻBDGLIT! ŠMGTHŠĒ ŅŅĢSURĒĢNPĀ! OT_M AZBTLJĻ, TPŪK VBĒŽOĢJ, UĻRUOĢJ ŪOIVJŪĪ! 25/05/2999
PMJŠŠ PŠVPJZCNT
ŅLIR_ ŅRUŅIŪĀTA
MKĢOZ UZŠMĢT_SŽ

ĀCDĒGHĪKLĢIJĶĻNORŠMŅPSTŪZ_ĀUVŽABČ
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
ĻBDGLIT! ŠMGTHŠĒ ŅŅĢSURĒĢNPĀ! OT_M AZBTLJĻ, TPŪK VBĒŽOĢJ, UĻRUOĢJ ŪOIVJŪĪ! 25/05/2999
PMJŠŠ PŠVPJZCNT
ŅLIR_ ŅRUŅIŪĀTA
MKĢOZ UZŠMĢT_SŽ

ĀCDĒGHĪKLĢIJĶĻNORŠMŅPSTŪZ_ĀUVŽABČ

Ievadiet atslēgu ==> 123456789

Teksts tika atšifrēts ar atslēgu 123456789:
LABDIEN! SLEPENA INFORMĀCIJA! ĻOTI SVARĪGI, ĻOTI SVARĪGI, PILNĪGI SLEPENI! 25/05/2999
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS

AĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ
```

3)

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> c

Nav šifrēts teksts:
LABDIEN! SLEPENA INFORMĀCIJA! ĻOTI SVARĪGI, ĻOTI SVARĪGI, PILNĪGI SLEPENI! 25/05/2999
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS

AĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ

Ievadiet atslēgu ==> 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
Kļūda! Ievadiet atslēgu kā ciparu virkni no 0 līdz 9, kas nav garāka par tekstu!
Ievadiet atslēgu ==> 1241235352

Teksts tika aizšifrēts ar atslēgu 1241235352:
ĻBEEJGS! TĻFTĒOC KSĢPŠPBDKMC! MRZĪ UĀCŪKĢJ, NSŽK TZBUJHK, UJĻOLĢJ VNHSĒOĶ! 25/05/2999
OKHMŠ RPTNIŠ_ŅŠ
ŅĪFNV SRŠMHRVOV
OJĒMŪ OSVNISŪŅŪ

CEDFEEFHGHĪĶKĻĶĶĻŅMŅPŠSUŠŠUZŪZ_BĀ
```

```
Ievadiet vai gribāt aizšifrēt (c) vai atšifrēt (d) tekstu ==> d

Šifrēts teksts:
ĻBEEJGS! TĻFTĒOC KSĢPŠPBDKMC! MRZĪ UĀCŪKĢJ, NSŽK TZBUJHK, UJĻOLĢJ VNHSĒOĶ! 25/05/2999
OKHMŠ RPTNIŠ_ŅŠ
ŅĪFNV SRŠMHRVOV
OJĒMŪ OSVNISŪŅŪ

CEDFEEFHGHĪĶKĻĶĶĻŅMŅPŠSUŠŠUZŪZ_BĀ

Ievadiet atslēgu ==> 1241235352

Teksts tika atšifrēts ar atslēgu 1241235352:
LABDIEN! SLEPENA INFORMĀCIJA! ĻOTI SVARĪGI, ĻOTI SVARĪGI, PILNĪGI SLEPENI! 25/05/2999
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS
LIELS NOSLĒPUMS

AĀBCČDEĒFGĢHIĪJKĶLĻMNŅOPRSŠTUŪVZŽ
```

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)

```
Teksts kurš ir ierakstīts teksts_to_morze.txt datnē un kuru var aizšifrēt:
MORSE CODE IS USEFUL

Teksts kurš ir ierakstīts morzes_kods.txt datnē un kuru var atšifrēt:
... ..- .-.. .- / .- .-. .. / .. .-. .- / .- .-.. ..- ...

Ja gribat atšifrēt vai aizšifrēt citu tekstu, tad izmainiet atbilstošu datnes saturu.

--------------------------------------------------------------------------------

Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> c

Teksts:
MORSE CODE IS USEFUL

Aizšifrēts teksts ar Morzes kodu:
-- --- .-. ... . / -.-. --- -.. . / .. ... / ..- ... . ..-. ..- .-..
```

2)

```
Teksts kurš ir ierakstīts teksts_to_morze.txt datnē un kuru var aizšifrēt:
MORSE CODE IS USEFUL

Teksts kurš ir ierakstīts morzes_kods.txt datnē un kuru var atšifrēt:
... ..- .-.. .- / .- .-. .. / .. .-. .- / .- .-.. ..- ...

Ja gribat atšifrēt vai aizšifrēt citu tekstu, tad izmainiet atbilstošu datnes saturu.

--------------------------------------------------------------------------------

Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> d

Aizšifrēts teksts ar Morzes kodu:
... ..- .-.. .- / .- .-. .. / .. .-. .- / .- .-.. ..- ...

Atšifrēts teksts:
SULA ARI IRA ALUS
```

3)

```
Teksts kurš ir ierakstīts teksts_to_morze.txt datnē un kuru var aizšifrēt:
HELLO EVERYONE I LOVE MATHEMATICAL ANALYSIS

Teksts kurš ir ierakstīts morzes_kods.txt datnē un kuru var atšifrēt:
... ..- .-.. .- / .- .-. .. / .. .-. .- / .- .-.. ..- ...

Ja gribat atšifrēt vai aizšifrēt citu tekstu, tad izmainiet atbilstošu datnes saturu.

--------------------------------------------------------------------------------

Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> sfa
Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> labi
Ievadiet vai gribāt aizšifrēt tekstu ar Morzes kodu (c) vai atšifrēt (d) tekstu no Morzes koda ==> C

Teksts:
HELLO EVERYONE I LOVE MATHEMATICAL ANALYSIS

Aizšifrēts teksts ar Morzes kodu:
.... . .-.. .-.. --- / . ...- . .-. -.-- --- -. . / .. / .-.. --- ...- . / -- .- - .... . -- .- - .. -.-. .- .-.. / .- -. .- .-.. -.-- ... .. ...
```

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)
```

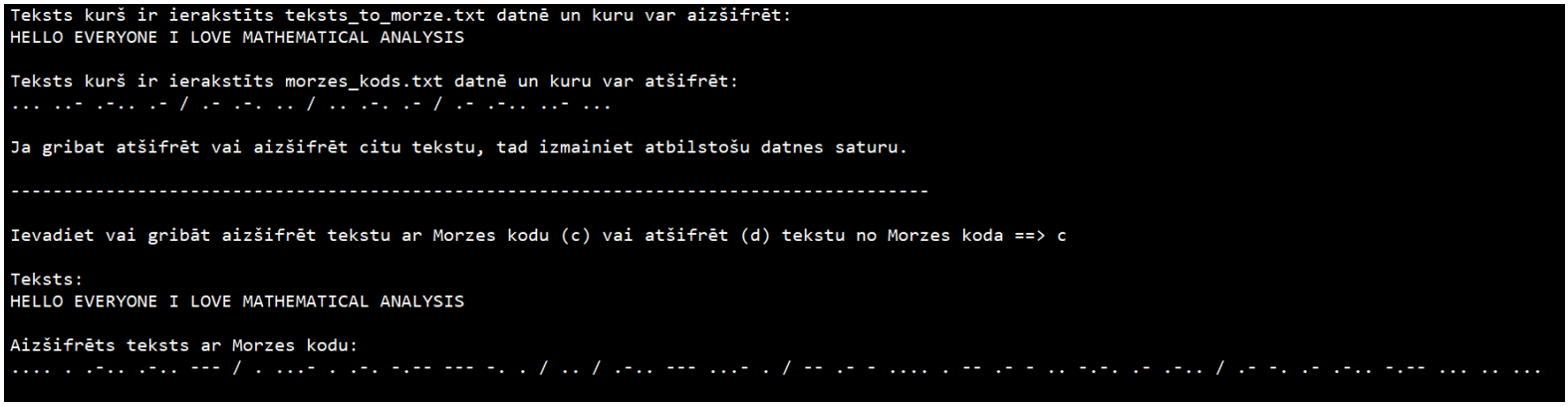
```
    print("\n\nAizšifrēts teksts ar Morzes kodu:")
    print(encrypted_text)  # Izvadīt aizšifrētu ar Morzes kodu tekstu.

    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:

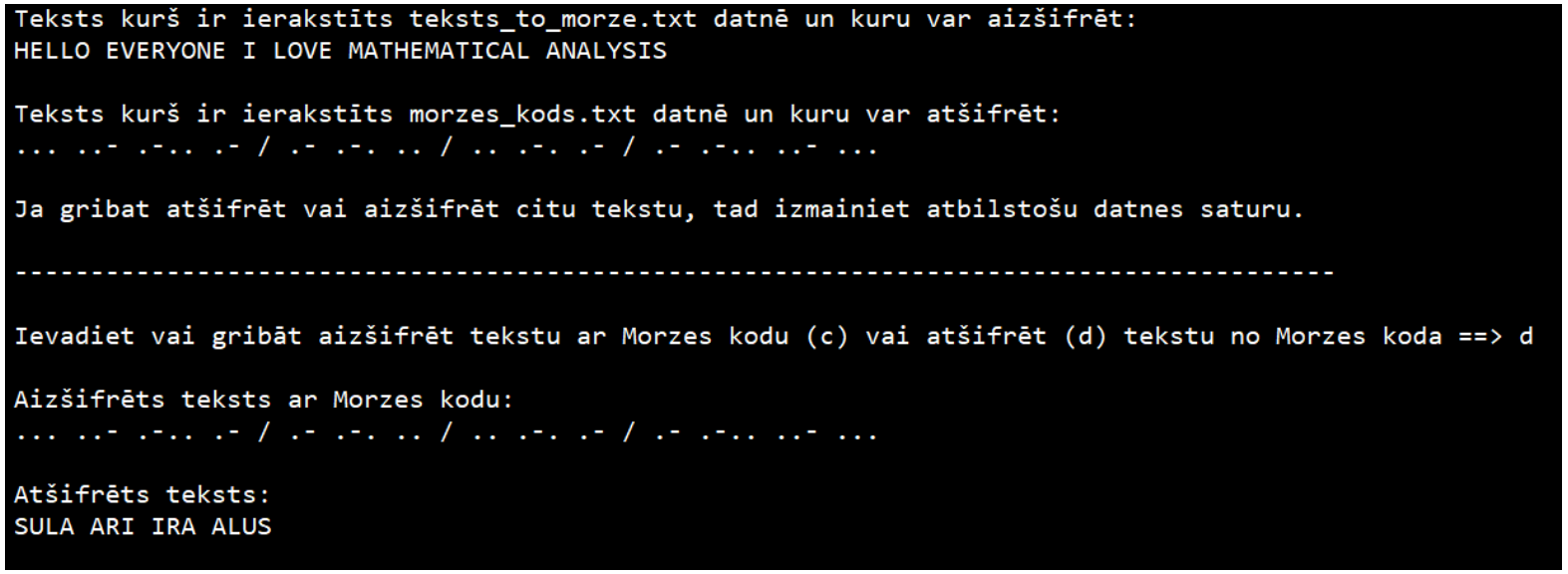
https://youtu.be/HbYfGPmI2Jg



2)

Video ar pikstēšanu:

https://youtu.be/YDbcTupV02E



3)

Video ar pikstēšanu:

https://youtu.be/qfoF6TskBJY