Задача:

Да се разработи стеганографска програма на езика Python.

1. Описание на програмата

При стартиране на програмата на екрана се появява меню, от което трябва да изберем действието, което искаме да се извърши:

1. Вмъкване на тайно съобщение (Encode)

2. Извличане на тайно съобщение (Decode)

```
from PIL import Image

print("     ************Python Kursova**************")

choice = input("""
                    1: Encode
                    2: Decode

                    Please enter your choice: """)
```

Тайното съобщение го разделяме на две части.

При избор на “1” програмата очаква въвеждането на името на текстовия файл, където искаме да бъде записано първата част от тайното съобщение (без разширение). Ако файлът не съществува на екрана се извежда съобщение за несъществуващ файл. Ако такъв файл съществува се преминава към следващата стъпка: въвеждане на съобщението.

```
if choice == "1":

    secretfile = input("Input the name of the file in which you want to hide a message (w/o extension): ")
    secretfile = secretfile + ".txt"

    try:
        f_crypt = open(secretfile, 'r')
        crypt_text = f_crypt.read()
        f_crypt.close()
    except FileNotFoundError:
        print("File does not exist!")
        quit()


    secrettext = input("Input the message you want to hide in the text file: ")
```

Криптирането на съобщението става като се извади 5 от ASCII кодовете на всеки символ от въведения низ. За целта трябва да дефинираме функция TextEncode, която извършва това:

```
def TextEncode(secrettext):
    encoded_text = []
    for letter in secrettext:
        for ch in letter:
            encoded_text.append(ord(ch) - 5)

    return ''.join(chr(ch) for ch in encoded_text)

TextEncode(secrettext)

f = open(secretfile,'w')
f.write(TextEncode(secrettext))
f.close()
```

Функцията има един параметър *secrettext*, в който се пази съобщението, което сме въвели. Преобразуването на съобщението става с помощта на променливата *encode_code*, която представлява празен списък, където ще добавим преобразуваните стойности. След като извадим 5 от ASCII кода на всеки символ от *secrettext*, функцията връща криптираното съобщение като низ и го записва във файла, който сме посочили.

Следващата стъпка е да въведем името на изображението, в което ще скрием втората част от съобщението. Отново правим проверка за съществуване на носещия файл. В променливата *imagesecret* се пази въведеното съобщение и се добавя символът “@” към него. Той служи за посочване на края на тайното съобщение и ще го използваме по-нататък при декодирането на съобщението.

```
secretimage = input("Input the name of the image in which you want to hide a message: ")

try:
    img = Image.open(secretimage)
except FileNotFoundError:
    print("Image does not exist!")
    quit()

imagesecret = input("Input the message you want to hide in the image: ") + '@'
```

Скриването на съобщението този път става като се добави 5 към ASCII кода на всеки символ от въведеното съобщение. Дефинираме функция ImageTextEncode, която е аналогична на функцията TextEncode.

```
def ImageTextEncode(imagesecret):
    img_new = []
    for letter in imagesecret:
        for ch in each:
            img_new.append(ord(ch) +5)

    return ''.join(chr(ch) for ch in img_new)

ImageTextEncode(imagesecret)
```

За вграждането на тайното съобщение в графичния контейнер ще използваме метода LSB. Същността на този метод е да замени последните значими битове в контейнера (в нашия случай изображение) с битовете на съобщението, което трябва да бъде скрито. За да постигнем това трябва да намерим двоичния запис на съобщението, което ще вмъкнем. За тази цел дефинираме функция EncodeTextToBin.

```
def EncodeTextToBin():
    secret = ImageTextEncode(imagesecret)
    img_bin = ""
    for symbol in secret:
        img_bin += str(format(ord(symbol), '08b'))
    return img_bin
EncodeTextToBin()
```

След това дефинираме функциите even и change, съответно за проверка за четност и промяна на най-младшия бит.

```
def even(num):
    if num % 2 == 0:
        return True
    else:
        return False

def change(num, bit):
    if bit == '0' and even(num):
        return num, 0
    if bit == '0' and not even(num):
        return num - 1, 1
    if bit == '1' and even(num):
        return num + 1, 1
    if bit == '1' and not even(num):
        return num, 0
```

Остава да вмъкнем тайното съобщение в изображението чрез функцията Embedding. В променливата *ImgBinText* записваме тайното съобщение в двоичен запис, а в *numberBytes* неговата дължината. Декларираме и променлива *br* за отброяване на всяко вмъкване и вземаме пикселната матрица на носещия файл, след което се извършва LSB модификация в изображението. В *maxByteImage* записваме максималният допустим размер на изображението, който се образува от хоризонталните пиксели, умножени по вертикалните, умножени по 3 (по 1 байт за цветовете). Дължината на тайното съобщението се получава от броя на символите умножени с 8, понеже за всеки символ ни трябват 8 байта.

```
def Embedding():
    ImgBinText = EncodeTextToBin()
    numberBytes = len(ImgBinText)
    br = 0
    pixels = img.load()
    changes = 0
    for i in range(0, img.size[0]):
        for j in range(0, img.size[1]):
            b,g,r = pixels[i,j]
            if br < numberBytes:
                b, ch = change(b, ImgBinText[br])
                br += 1
                changes += ch
            if br < numberBytes:
                g, ch = change(g, ImgBinText[br])
                br += 1
                changes += ch
            if br < numberBytes:
                r, ch = change(r, ImgBinText[br])
                br += 1
                changes += ch
            pixels[i,j] = b,g,r

    maxByteImage = img.size[0]*img.size[1]*3
    lengthSecretText = len(imagesecret)*8
```

Накрая остава да проверим дали съобщението може да се вгради в носещия файл. Ако дължината на тайното съобщение е по-голяма от размера на носещия файл на екрана се отпечатва съобщение, че тайното съобщение е твърде голямо за вмъкване. В противен случай стегофайлът се записва с име “encoded” в текущата папка, отпечатват се броят промени, необходими за вмъкването на съобщението, както и съобщение за успешно вмъкване.

```
if lengthSecretText > maxByteImage:
    print("The message is too big for the image! ")
    quit()
else:
    img.save("encoded.bmp")
    img.close()
    print("Total changes needed to hide the message:", changes)
    print("Succesfully encoded! ")
```

При избор на “2” от менюто програмата очаква въвеждането на името на текстовия файл, в който е записано тайно съобщение и името на стегофайла като отново се прави проверка за съществуване.

```
elif choice == "2":
    secretfile = input("Input the name of the file which you want to decode (w/o) extension: ")
    secretfile = secretfile + ".txt"

    try:
        f_crypt = open(secretfile,'r')
        crypt_text = f_crypt.read()
        f_crypt.close()
    except FileNotFoundError:
        print("File does not exist!")
        quit()

    secretimage = input("Input the name of the image you want to decode (w/o extension): ")
    secretimage = secretimage + ".bmp"

    try:
        img = Image.open(secretimage)
        pixels = img.load()
        l_pixels = []
    except FileNotFoundError:
        print("Image does not exist!")
        quit()
```

При криптирането на първата част от съобщението извадихме 5 от ASCII кода на всеки символ. За да декодираме съобщението дефинираме функция DecodeText, която добавя 5 към ASCII кода на всеки символ във файла.

```
def DecodeText(crypt_text):
    new_text = []
    for letter in crypt_text:
        for ch in letter:
            new_text.append(ord(ch) + 5)
    return ''.join(chr(ch) for ch in new_text)

DecodeText(crypt_text)
```

При вмъкването на втората част от съобщението в изображението добавихме 5 към ASCII кода на всеки символ. За да декодираме съобщението трябва да направим следното:

- дефинираме функция Getsymbol, която намира двоичния вид на най-младшия бит и връща символа, съответстващ на неговия ASCII код, като изважда 5 от съответната стойност;
- дефинираме функция PixelMatrixToList, която връща пикселната матрица като списък;

- в променлива *stegoText* записваме всеки символ, който декодирахме чрез функцията Getsymbol до срещане на символа “@” ;

```
def Getsymbol(octet):
    bnr = ""
    for each in range(0,len(octet)):
        bnr += str(format(octet[each],'08b'))[7]
    return chr(int(bnr,2) -5)


def PixelMatrixToList(pixels):
    for i in range(0,img.size[0]):
        for j in range(0,img.size[1]):
            b,g,r = pixels[i,j]
            l_pixels.append(b)
            l_pixels.append(g)
            l_pixels.append(r)
    return l_pixels
PixelMatrixToList(pixels)

stegoText = ""
for i in range(0,len(l_pixels),8):
    symbol = Getsymbol(l_pixels[i:i+8])
    if symbol == '@':
        break
    else:
        stegoText += symbol
```

При избор на грешно действие се отпечатва съобщение за грешен избор.

```
else:
    print("Wrong choice!")
    print("Please try again!")
```

2. Тестване на програмата

Използвани са следните графични контейнери:

Графичен контейнер 1:

- формат .bmp, 24-битова растерна графика, 60x60 пиксела, размер 11 KB

Графичен контейнер 2:

- формат .jpg, 24 битов цвят, 500x500 пиксела, 176 т/инч, размер 111 KB

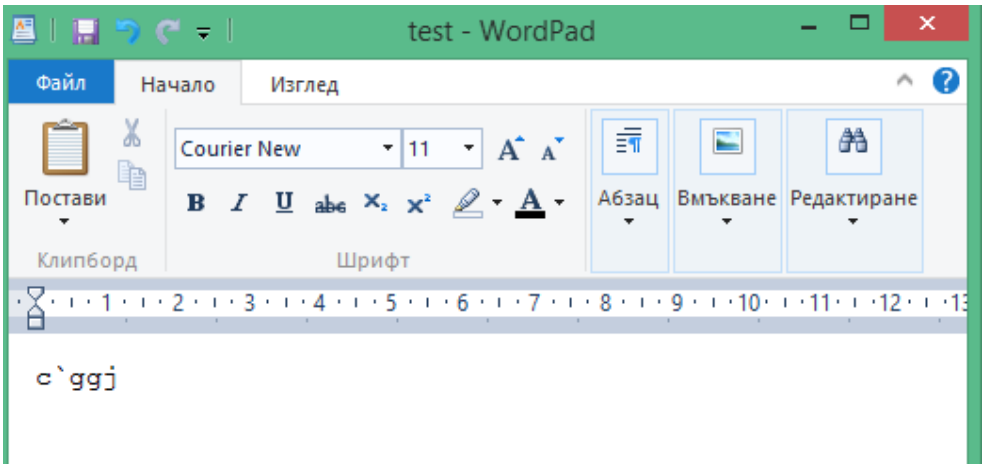
2.1. Тестване посредством графичен контейнер 1 (Encoding):

```
HP@G4 C:\Users\HP\Desktop\python kursova
$ python kursova.py
            ************Python Kursova**************


                    1: Encode
                    2: Decode

                    Please enter your choice: 1
Input the name of the file in which you want to hide a message (w/o extension): test
Input the message you want to hide in the text file: hello
Input the name of the image in which you want to hide a message: test.bmp
Input the message you want to hide in the image: world
Total changes needed to hide the message: 24
Succesfully encoded!
```

Проверяваме дали първата част от тайното съобщение е записано в текстовия файл:



Графичен контейнер (test.bmp)

 размер – 11 KB

Стегофайл (encoded.bmp)

 размер – 11 KB

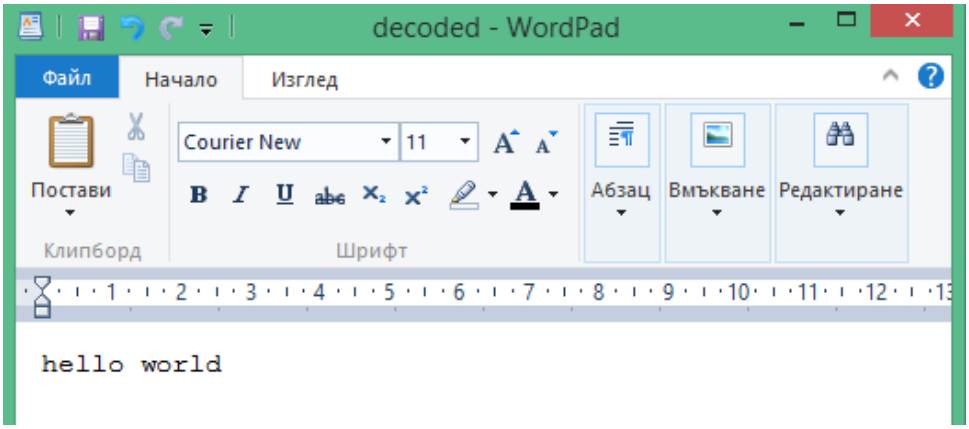
2.2. Тестване посредством графичен контейнер 1 (Decoding):

```
HP@G4 C:\Users\HP\Desktop\python kursova
$ python kursova.py
             *************Python Kursova***************


                    1: Encode
                    2: Decode

                    Please enter your choice: 2
Input the name of the file which you want to decode (w/o extension:) test
Input the name of the image you want to decode (w/o extension): encoded
The hidden message is: hello world
```

Проверяваме дали декриптираното съобщение е записано в decoded.txt:



2.3. Тестване посредством графичен контейнер 2 (Encoding):

```
HP@G4 C:\Users\HP\Desktop\python kursova
$ python kursova.py
            ************Python Kursova**************


                    1: Encode
                    2: Decode

                    Please enter your choice: 1
Input the name of the file in which you want to hide a message (w/o extension): test
Input the message you want to hide in the text file: hello
Input the name of the image in which you want to hide a message: test2.jpg
Input the message you want to hide in the image: world
Total changes needed to hide the message: 25
Succesfully encoded!
```

Графичен контейнер (test2.jpg)

 размер – 111 KB

Стегофайл (encoded.bmp)

 размер – 733 KB

3. Изводи

Изводите, които могат да се направят вследствие от извършените експерименти, са:

- Не се наблюдават визуални промени в изображенията;
- Размерите на графичния контейнер и стегофайла са идентични при първи експеримент;
- Размерите на графичния контейнер и стегофайла са различни при втори експеримент;
- Промените, които се извършат за вмъкване на тайното съобщение са повече при графичен контейнер 2.