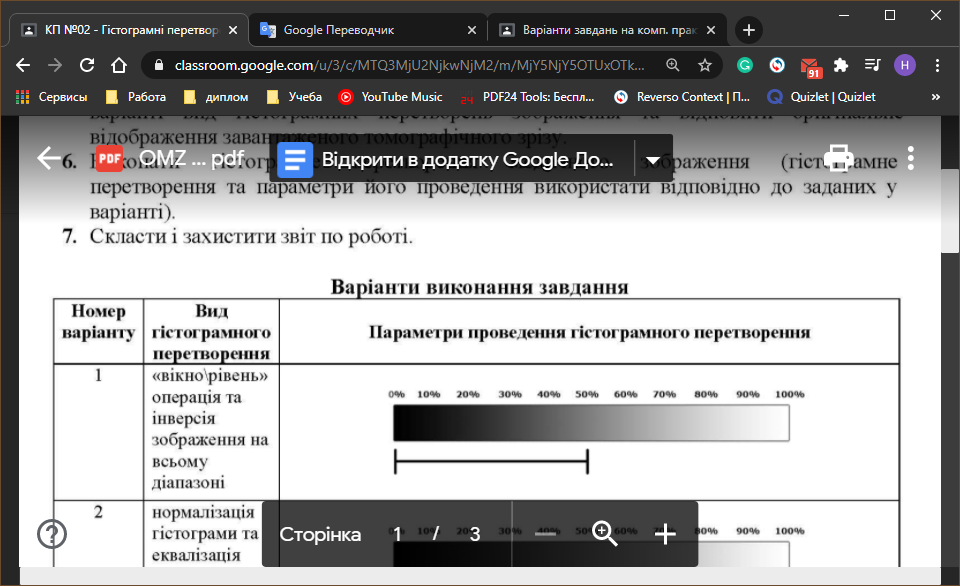
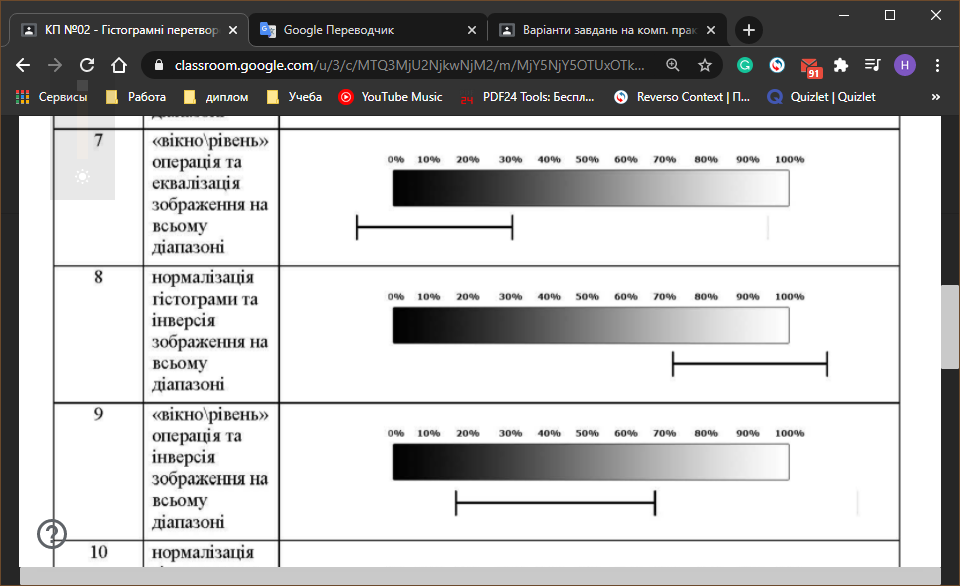
**Завдання**

1. Ознайомитися з теоретичними відомостями гістограмних перетворень.
2. Розробити програмний додаток для завантаження мадичного зображення в форматі DICOM та виконання гістограмних перетворень.
3. Розміри частини вікна програмного додаткі для візуалізації графічних даних (без інтерфейсу користувача) мають відповідати розмірам завантаженого медичного зображення; завантажене медичне зображення має мати масштаб 100% (одному пікселю зображення відповідає один піксель екрана).
4. Реалізувати інтерактивне текстове відображення у вінкі програмного додатка значення яскравості пікселя під курсором лише під час переміщення миші на даних завантаженго зображення. Після виконання гістограмного перетворення має виконуватися відповідна зміна значень яскравості, що відображається.
5. Створити події від клавіатури, при обробці яких можна окремо виконати заданий у варіанті вид гістограмних перетворень зображення та відновити оригінальне відображення завантаженого томографічного зрізу.
6. Виконати гістограмне перетворення медичного зображення (гістограмне перетворення та параметри його проведення використати відповідно до заданих у варіанті).
7. Скласти і захистити звіт по роботі.



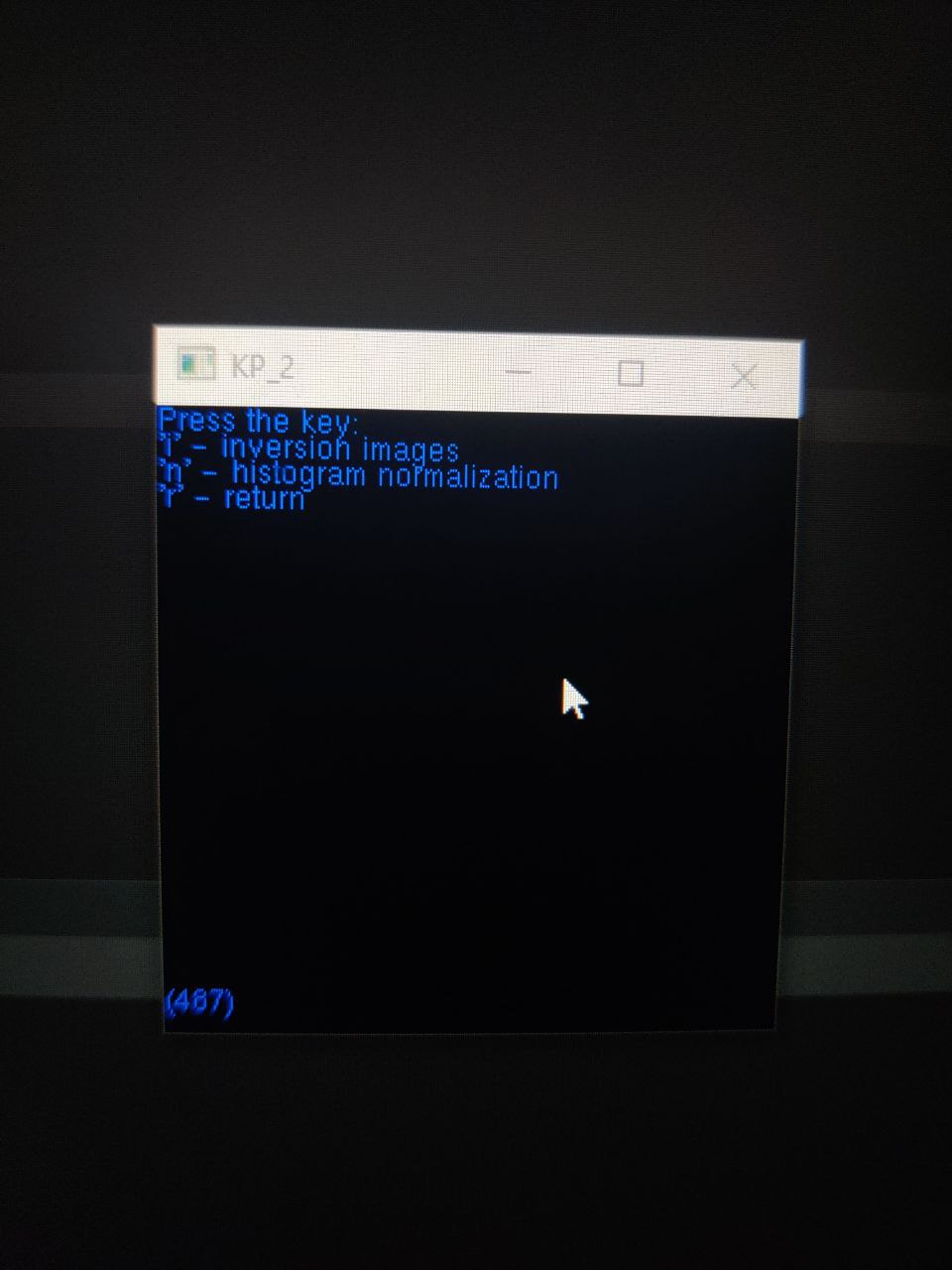


**Лістинг программи**

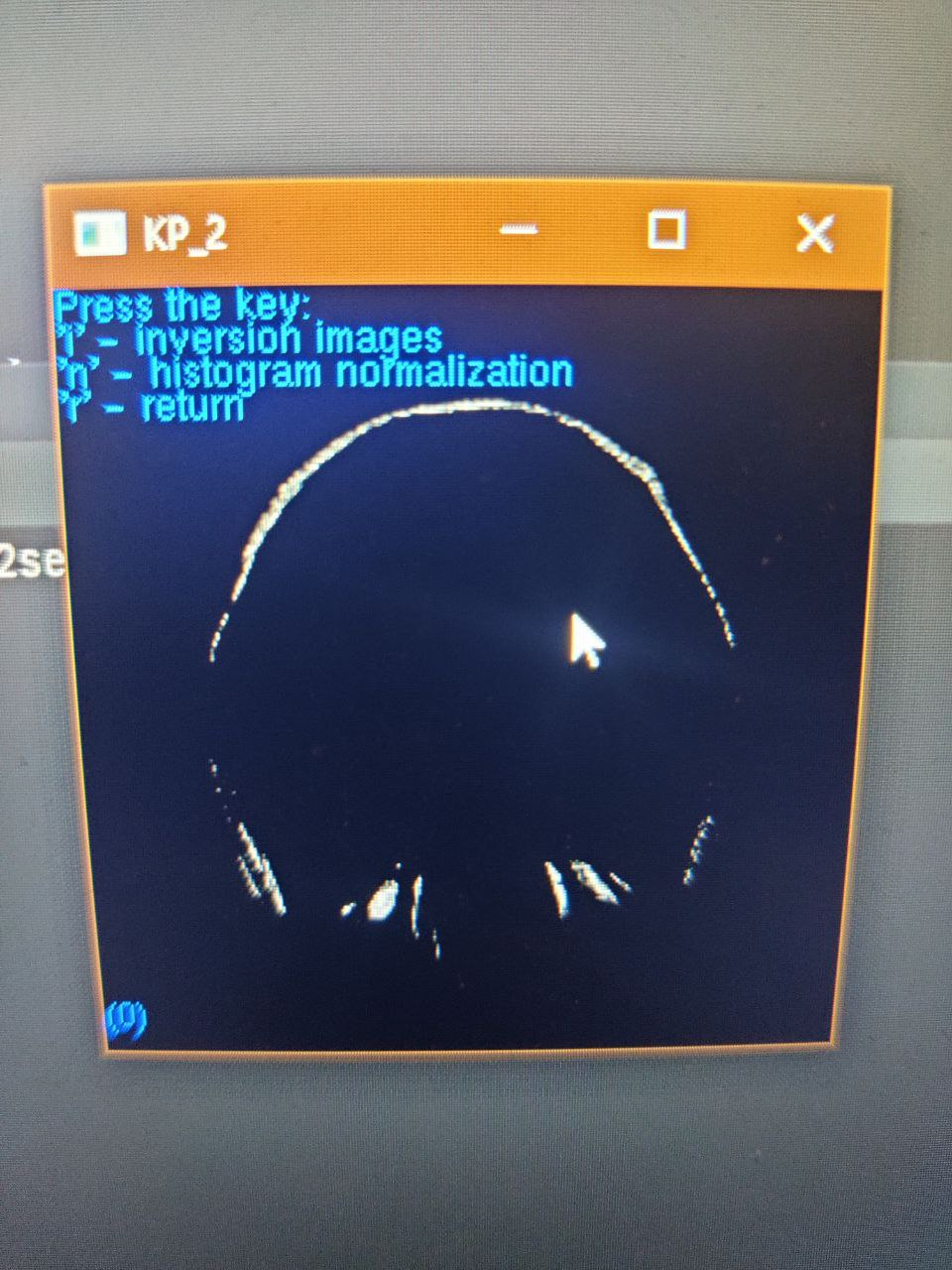
from OpenGL.GL import \*  
from OpenGL.GLU import \*  
from OpenGL.GLUT import \*  
import pydicom  
import numpy as np  
  
path\_file = "DICOM\_Image\_16b.dcm"  
  
  
class Image:  
 def \_\_init\_\_(self, path):  
 self.ds = pydicom.read\_file(path)  
 self.image\_pixels = np.array(self.ds.pixel\_array)  
 self.width, self.height = self.ds[0x280010].value, self.ds[0x280011].value  
  
 self.mouse\_on\_image = False  
 self.x\_coord, self.y\_coord = 0, 0  
  
 self.is\_normalization = False  
 self.is\_inversion = False  
  
 def init(self):  
 glClearColor(0, 0, 0, 0.0)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity()  
 gluOrtho2D(0, self.width, 0, self.height)  
  
 def display(self):  
 glClear(GL\_COLOR\_BUFFER\_BIT)  
 glColor3f(1, 1, 1)  
  
 draw\_pixel = self.image\_pixels  
 if self.is\_normalization:  
 draw\_pixel = self.normalization\_func(draw\_pixel, 0.75, 1)  
 elif self.is\_inversion:  
 draw\_pixel = self.inversion\_func(draw\_pixel)  
  
 self.draw\_texture(draw\_pixel, GL\_LUMINANCE)  
  
 if self.mouse\_on\_image:  
 self.print\_text(0, 10, GLUT\_BITMAP\_HELVETICA\_12, "(" + str(draw\_pixel[self.x\_coord][self.height - self.y\_coord]) + ")")  
  
 self.print\_text(0, self.height - 40, GLUT\_BITMAP\_HELVETICA\_12, "'r' - return")  
 self.print\_text(0, self.height - 30, GLUT\_BITMAP\_HELVETICA\_12, "'n' - histogram normalization")  
 self.print\_text(0, self.height - 20, GLUT\_BITMAP\_HELVETICA\_12, "'i' - inversion images")  
 self.print\_text(0, self.height - 10, GLUT\_BITMAP\_HELVETICA\_12, "Press the key:")  
  
 glutSwapBuffers()  
  
 def print\_text(self, x, y, font, line):  
 glColor3f(0, 0, 1)  
 glPushAttrib(GL\_DEPTH\_TEST)  
 glRasterPos2d(x, y)  
 for i in line:  
 glutBitmapCharacter(font, ord(i))  
 glPopAttrib()  
  
 def draw\_texture(self, data, internal\_format):  
 glTexImage2D(GL\_TEXTURE\_2D, 0, internal\_format, self.width, self.height, 0, internal\_format, GL\_UNSIGNED\_BYTE, data)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST)  
 glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST)  
 glEnable(GL\_TEXTURE\_2D)  
  
 glBegin(GL\_QUADS)  
 glTexCoord2d(0.0, 0.0)  
 glVertex2d(0.0, 0.0)  
 glTexCoord2d(1.0, 0.0)  
 glVertex2d(self.width, 0.0)  
 glTexCoord2d(1.0, 1.0)  
 glVertex2d(self.width, self.height)  
 glTexCoord2d(0.0, 1.0)  
 glVertex2d(0.0, self.height)  
 glEnd()  
  
 glDisable(GL\_TEXTURE\_2D)  
  
 def normalization\_func(self, pixels, p\_min, p\_max):  
 p\_min, p\_max = p\_min \* pixels.max(), p\_max \* pixels.max()  
 normalization = p\_min + (pixels - pixels.min()) / (pixels.max() - pixels.min()) \* (p\_max - p\_min)  
  
 return normalization.astype(int)  
  
 def inversion\_func(self, pixels):  
 inversion = pixels.max() - pixels + pixels.min()  
 return inversion  
  
 def pixels\_transformations(self, data):  
 transform\_image = []  
 for i in range(self.width):  
 transform\_image.append([])  
 for j in range(self.height):  
 transform\_image[i].append(data[i][j] and self.image\_pixels[i][j])  
  
 return transform\_image  
  
 def create\_gradient(self):  
 gradient = {}  
 color = 0  
 for key in range(np.amin(self.image\_pixels), np.amax(self.image\_pixels) // 2):  
 gradient[key] = color  
 color += 1  
 if color > 255:  
 color = 255  
 for key in range(np.amax(self.image\_pixels) // 2, np.amax(self.image\_pixels) + 1):  
 gradient[key] = color  
 color -= 1  
 if color < 0:  
 color = 0  
  
 gradient\_image = []  
 for pixel\_line in self.image\_pixels:  
 line = []  
 for pixel in pixel\_line:  
 line.append(gradient[pixel])  
 gradient\_image.append(line)  
  
 return np.array(gradient\_image)  
  
 def channel\_G(self, data):  
 rgb = np.zeros((self.height, self.width, 3), 'uint8')  
  
 rgb[..., 1] = data  
 rgb[..., 0] = 0  
 rgb[..., 2] = 0  
  
 return rgb  
  
 def keyboard\_func(self, my\_key, x, y):  
 key = unicode(my\_key, errors='ignore')  
 if key == 'n':  
 self.is\_normalization = True  
 self.is\_inversion = False  
 if key == 'i':  
 self.is\_inversion = True  
 self.is\_normalization = False  
 if key == 'r':  
 self.is\_inversion = False  
 self.is\_normalization = False  
  
 def motion\_func(self, x, y):  
 self.x\_coord, self.y\_coord = x, y  
 if (0 < x < self.width) and (0 < y < self.height):  
 self.mouse\_on\_image = True  
 else:  
 self.mouse\_on\_image = False  
 self.display()  
  
  
def init\_window(width, height):  
 glutInitWindowSize(width, height)  
 glutInitWindowPosition((glutGet(GLUT\_SCREEN\_WIDTH) - width) // 2, (glutGet(GLUT\_SCREEN\_HEIGHT) - height) // 2)  
 glutCreateWindow('KP\_2')  
  
  
def main():  
 glutInit()  
 glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)  
 file = Image(path\_file)  
 init\_window(file.width, file.height)  
 file.init()  
 glutDisplayFunc(file.display)  
 glutKeyboardFunc(file.keyboard\_func)  
 glutPassiveMotionFunc(file.motion\_func)  
 glutMainLoop()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Результат

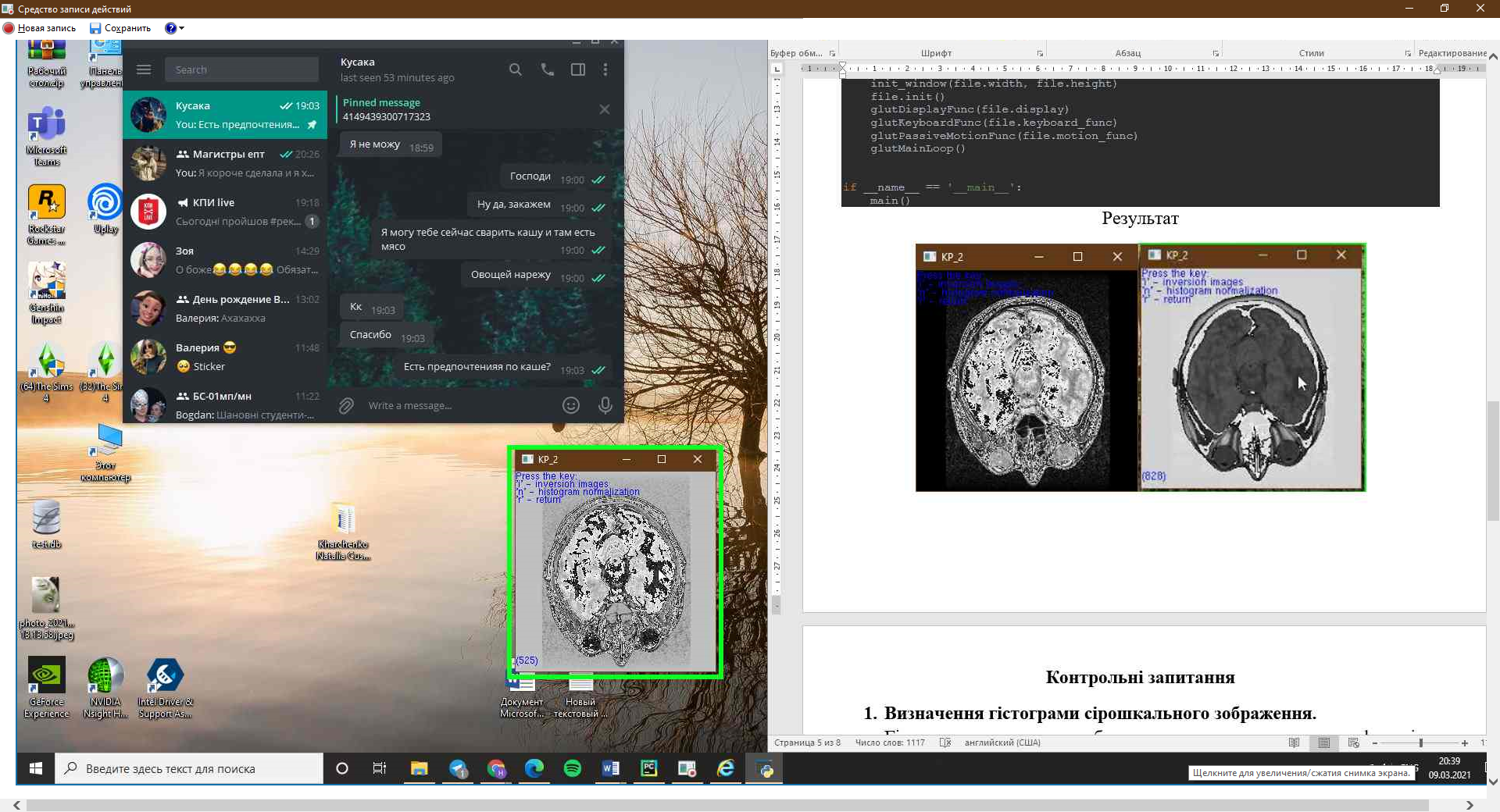
Початкове зображення



Норалізація



Інверсія



**Контрольні запитання**

1. **Визначення гістограми сірошкального зображення.**

Гістограма дискретного зображення називажться дискретна функція , де – кількість рікселів, які мають яскравість , а – кількість пікселів у всьому зображенні. Значення є оцінкою ісовірності появи пікселя яскравості в зображенні.

При створенні гістограми для сірошкального зображення вона буде виглядати як двовимірний графік розподілу сірих відтінків в зображенні. Чим правіше розташована точка на горизонтальній осі, тим яскравішому пікселю вона відповідаю.

Гістограма манохромного зображення – це масив з 256 елементів кожен елемент якого – кількість пікселів з відповідною яскравістю.

1. **Що являє собою гістограма кольорового зображення?**

Гістограмма кольорового зображення будується по трьом колірним каналам (червоний, зелений, синій) та виводиться три графіки.

Отже вона має три масиви з 256 елементів, де колен відповідає кількості пікселів певної яскравісті відповідно до кольору.

1. **Що таке нормована гістограма, навіщо вона потрібна?**

Нормована гістограмма представляє значення у вигляді стовпчиків з накопиченням для представлення 100%. Її можна використовувати коли присутне декілька рядів даних та необхідно підкреслити їх вклад у загальне значення, особливо коли загальне значення однакове для всіх категорій.

Нормалізація гістограми представляє процесс розтягнення певного піку на діапазон рівнів яскравості.

1. **В чому полягають особливості програмно-алгоритмічної реалізації розрахунку гістограми, кумулятивної гістограми?**

При проведені розрахунків мінімальній яскравості відповідає найправіша точка (що відповідає кумулятивній гістограмі, стовпці впорядковані за зростанням), також висота кожного стовпчика дорівнює числу пікселів зображення, які мають відповідну яскравість.

1. **Що таке гістограмні перетворення? Їх основні види.**

Гістограмні перетворення це зміна початкового зображення певним чином, аби гістограма яскравості набула необхідної форми.  
Види гістограм:

* Вікно/рівень;
* Нормалізація гістограми;
* Бінаризація;
* Інверсія;
* Еквалізація.

1. **Як зміна «рівня» та «ширина вікна» впливають на значення яскравості та контрастності зображення?**

Розтягнення тонового діапазону збільшує контастність зображення.

1. **В чому полягає різниця, а в чому схожість проведення «вікно/рівень» операції та нормалізації гістограми щодо обраного піку яскравостей?**

Схожість полягає у тому, що в обох методах ми проводимо виміри у певному діапазоні значень. Різниця: у тому що при викоонанні операції «вінко/рівень» пікселі які не потравляють у діапазон замінюються на мінмум або максимум.

1. **Чому еквалізація зображення працює краще для даних без фону зображення?**

Еквалізація – це вирівнювання гістограми, тобто проведення таких перетворень, щоб на зображені у рівній кількості були присутні пікселі з різними значеннями із заданого динамічного діапазону інтенсивності.

Коли на зображені присутній фон, то ми отримуємо додаткові значення, що спотворюють зображення при еквалізації, тому що враховуються значення фону. Коли зображення без фону, то фонові пікселі мають значення нуль, що не впливає на обробку.

1. **Як визначити максимальне значення яскравості з діапазону рівнів яскравості зображення для формули кожного із гістограмних перетворень? На що це значення впливає?**

Максимальне значення яскравості рівне значенню максимальної яскравості пікселі зображення. Вихідне значення яскравості буде залежне від типу перетворень.

1. **Що визначає двовимірна гістограма сірошкального зображення?**

Вона визначає кількість пікселів певної якравості (з права на ліво – від найяскравішої точки до найтемнішої). Висота кожного стовпчика відповідає кількості пікселів певної яскравості.