MỤC LỤC

Hồ Phúc Lâm 22680401

Đề cương ôn tập

1. Phần Audio
    a. Đọc file
    b. Hiển thị thuộc tính
    c. Biểu diễn vẽ tín hiệu Audio
    d. Khác
2. Phần Image và video
    a. Đọc file
    b. Hiển thị các thuộc tính
    c. Các hàm chỉnh sửa ảnh
    d. Khác
3. Yêu cầu SV tạo thư mục Data chứ 5 file Audio, video và 5 file image. Cài đặt trên laptop cá nhân

Môn : Phát triển hệ thống đa phương tiện DHCNTT18ABTT

Câu 1:

a)Cho Code Point U+00E6, U+00FC. Hãy xác định mã UTF-8 2 bytes tương ứng.

b) Mã utf-8 của "Đ" là 0xc4,0x90; "Ạ" là 0xe1,0xba,0xa0; "Ọ" là 0xe1,0xbb,0x8c; "Ô" là 0xc3,0x94. "Ệ" là 0xe1,0xbb,0x86. A/C hãy viết code Python giải mã để cho ra kết quả là "ĐẠI HỌC CÔNG NGHIỆP"

Câu 2:

a) Việc chuyển đổi từ tín hiệu Analog sang Digital với mục đích gì ?Trình bày quá trình chuyển đổi tín hiệu từ Analog sang Digital. b)Viết chương trình đọc file Audio và hiển thị thông tin về file. c)Viết chương trình tạo tín hiệu Audio hình Sin hoặc Cosin với các tham số liên quan như tần số lấy mẫu, tần số tín hiệu, biên độ (các tham số này tự chọn)

Câu 3:

a)A/C trình bày các mô hình màu RGB, YIQ, YUV và YCrCb. b)Dữ liệu video có tốc độ truyền 24 fps, mỗi frame có kích thước 720x486, mô hình YCrCb chroma subsampling sử dụng là 4:2:2.Xác định dung lượng lưu trữ video trong 1 phút 15 giây.

Câu 4:

a)Viết chương trình hiển thị nội dung ảnh có định dạng jpg, png, bitmap b)Viết chương trình thực hiện các chức năng biến đổi ảnh như : làm mờ ảnh

c) Viết chương trình thực hiện các chức năng biến đổi ảnh như xoay ảnh theo 1 góc tùy ý (góc <=20 độ)

d) Viết chương trình thực hiện các chức năng biến đổi ảnh như co giãn ảnh.

e) Viết chương trình thực hiện các chức năng biến đổi ảnh như lật ảnh từ trái sang phải(hoặc ngược lại)

f) Viết chương trình thực hiện các chức năng biến đổi ảnh như lật ảnh từ trên xuống dưới(hoặc ngược lại).

Câu 5:Viết chương trình phát hiện khuôn mặt người trên ảnh.

THỰC HIỆN

Câu 1:

a)Cho Code Point U+00E6, U+00FC. Hãy xác định mã UTF-8 2 bytes tương ứng.

Truy cập https://www.utf8-chartable.de/unicode-utf8-table.pl

Unicode code point	character	UTF-8 (hex.)	name
U+00E6	æ	c3 a6	LATIN SMALL LETTER AE
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS

Nhắc lại kiến thức và xem lại LT02:

Thập phân	Nhị phân (4bits)	Thập lục (hex)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

a1) Cho Code Point U+00E6

Bước 1: Viết Code Point dưới dạng chuỗi nhị phân

U+00E6 => 00E6 => 0000 0000 1110 $0110_{(2)}$

Vì cần chuyển đổi sang dạng UTF-8 2 bytes nên:

Lấy 11 bits từ bên phải sang bên trái

0000 0000 1110 0110 => 000 1110 0110

Tiếp theo, chia thành 2 phần gồm I 5 bits cao xét bên trái và II 6 bits thấp xét bên phải

000 1110 0110

Bước 2: đưa xuống bảng

1	1	0	0	0	0	1	1
1	0	1	0	0	1	1	0

Bước 3: đáp số

C	3
A	6

C3 A6

Đối chiếu kết quả

Unicode code point	**character**	**UTF-8 (hex.)**	**name**
U+00E6	æ	c3 a6	LATIN SMALL LETTER AE

a2)Cho Code Point U+00FC

Bước 1: Viết Code Point dưới dạng chuỗi nhị phân

U+00FC => 00FC => 0000 0000 1111 $1100_{(2)}$

Vì cần chuyển đổi sang dạng UTF-8 2 bytes nên:

Lấy 11 bits từ bên phải sang bên trái

0000 0000 1111 1100 => 000 1111 1100

Tiếp theo, chia thành 2 phần gồm I 5 bits cao xét bên trái và II 6 bits thấp xét bên phải

000 1111 1100

Bước 2: đưa xuống bảng

1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	0

Bước 3: đáp số

C	3
B	C

C3 BC

Đối chiếu kết quả

Unicode code point	**character**	**UTF-8 (hex.)**	**name**
U+00FC	ü	c3 bc	LATIN SMALL LETTER U WITH DIAERESIS

b) Mã utf-8 của "Đ" là 0xc4,0x90; "Ạ" là 0xe1,0xba,0xa0; "Ọ" là 0xe1,0xbb,0x8c; "Ô" là 0xc3,0x94. "Ệ" là 0xe1,0xbb,0x86. A/C hãy viết code Python giải mã để cho ra kết quả là "ĐẠI HỌC CÔNG NGHIỆP"

```
# 1. Danh sách hex values chứa các mã hex UTF-8 của từng ký tự
hex_values = [
    [0xc4, 0x90],  # Đ
    [0xe1, 0xba, 0xa0],  # Ạ
    [0x49],  # I (ký tự 'I' không phải mã hóa đặc biệt)
    [0x20],  # (space)
    [0x48],  # H
    [0xe1, 0xbb, 0x8c],  # Ọ
    [0x43],  # C
    [0x20],  # (space)
    [0x43],  # C
    [0xc3, 0x94],  # Ô
    [0x4e],  # N
    [0x47],  # G
    [0x20],  # (space)
    [0x4e],  # N
    [0x47],  # G
    [0x48],  # H
    [0xe1, 0xbb, 0x86],  # Ệ
    [0x50]   # P
]
# 2. Giải mã từng nhóm hex thành ký tự UTF-8
decoded_string = ''.join([bytes(item).decode('utf-8') for item in hex_values])

# 3. In ra chuỗi đã giải mã
print(decoded_string)
```

Câu 2:

a) Việc chuyển đổi từ tín hiệu Analog sang Digital với mục đích gì ?Trình bày quá trình chuyển đổi tín hiệu từ Analog sang Digital.

- Chuyển đổi tín hiệu Analog (tín hiệu tương tự, liên tục) sang tín hiệu Digital. Vì máy tính xử lý thông tin rời rạc (gồm 0 và 1) trong nên cần phải chuyển tín hiệu liên tục analog sang tín hiệu số Digital.

- Giúp lưu trữ và truyền tải tín hiệu. Dễ xử lý và phân tích bằng thuật toán. Ít bị nhiễu và suy giảm hơn khi truyền qua các kênh khác nhau. Bảo toàn dữ liệu và sao lưu, phục hồi tín hiệu gốc.

Để chuyển đổi tín hiệu Analog sang Digital, người ta thực hiện qua 3 giai đoạn: Sampling, Quantization và Coding

- **Sampling (Lấy mẫu):** Tín hiệu analog thực tế (như nhiệt độ, độ ẩm, âm thanh,…) được lấy mẫu theo các khoảng thời gian nhất định. Mỗi khoảng thời gian, tín hiệu âm thanh sẽ được đo và lưu trữ giá trị của nó
- **Quantization (Lượng tử hóa):** Các giá trị mẫu âm thanh được chuyển đổi thành các trạng thái 0 và 1.
    - Ví dụ, tín hiệu có giá trị mẫu thấp hơn ngưỡng được mã hóa thành 0, trong khi tín hiệu có giá trị mẫu cao hơn ngưỡng được mã hóa thành 1.
- **Coding (Mã hóa):** Tín hiệu số được biểu diễn bằng các bit. Quá trình này cho phép tín hiệu âm thanh được lưu trữ, truyền và tái tạo lại một cách chính xác.

Kỹ thuật PCM (Pulse-Code Modulation) là một phương pháp biểu diễn kỹ thuật số của tín hiệu analog mà sẽ đưa các mẫu biên độ của tín hiệu analog đều đặn. Dữ liệu tương tự được lấy mẫu được thay đổi thành, và sau đó được biểu thị bằng dữ liệu nhị phân. PCM yêu cầu một chiếc đồng hồ rất chính xác.

Ví dụ về kỹ thuật PCM Coding: Trong công nghệ âm thanh, PCM được sử dụng rộng rãi. Ví dụ, khi bạn ghi âm bằng phần mềm ghi âm trên máy tính hoặc điện thoại, tín hiệu âm thanh từ micro sẽ được chuyển đổi từ dạng analog sang dạng số bằng kỹ thuật PCM. Sau khi ghi xong, bạn có thể lưu lại file âm thanh dưới dạng số này để nghe lại sau.

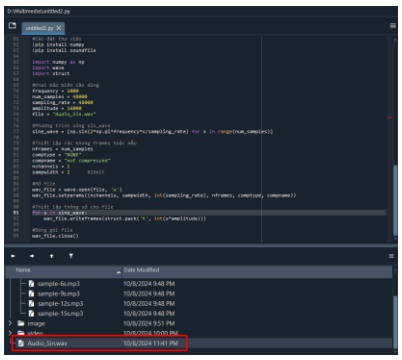
b)Viết chương trình đọc file Audio và hiển thị thông tin về file.

```
#Cài đặt thư viện mutagen
!pip install mutagen

from mutagen.mp3 import MP3
def show_mp3_properties(file_path):
    # Load the MP3 file
    audio = MP3(file_path)

    # Get properties
    duration = audio.info.length  # Duration in seconds
    bitrate = audio.info.bitrate / 1000  # Bitrate in kbps
    sample_rate = audio.info.sample_rate  # Sample rate in Hz
    channels = audio.info.channels  # Number of channels

    # Print properties
    print(f"File: {file_path}")
    print(f"Duration: {duration:.2f} seconds")
    print(f"Bitrate: {bitrate} kbps")
    print(f"Sample Rate: {sample_rate} Hz")
    print(f"Channels: {channels}")

# Example usage
file_path = "D:/Multimedia/Data/audio/sample-3s.mp3"
show_mp3_properties(file_path)
```

```
File: D:/Multimedia/Data/audio/sample-3s.mp3
Duration: 3.24 seconds
Bitrate: 127.998 kbps
Sample Rate: 44100 Hz
Channels: 2
```

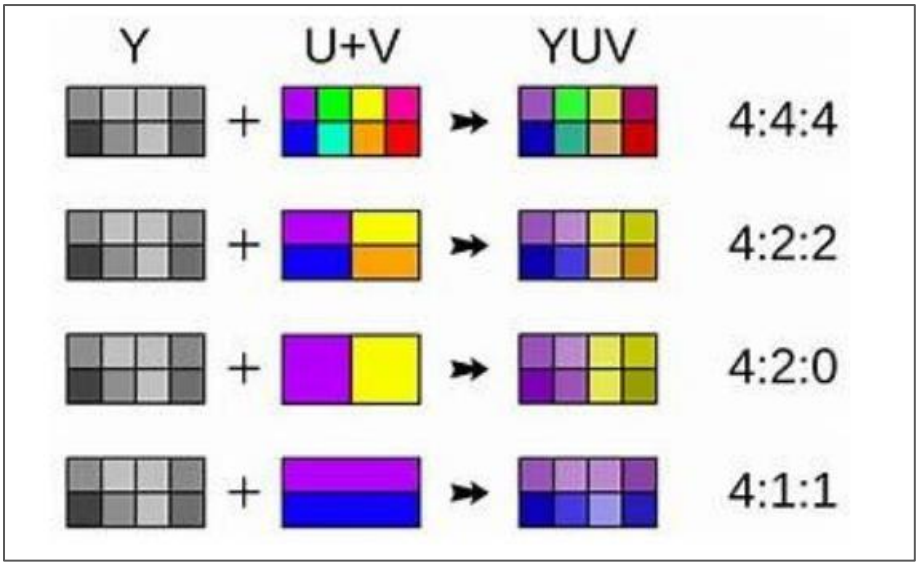
c)Viết chương trình tạo tín hiệu Audio hình Sin hoặc Cosin với các tham số liên quan như tần số lấy mẫu, tần số tín hiệu, biên độ (các tham số này tự chọn)

```
import numpy as np
import wave
import struct

#Khai báo biến cần dùng
frequency = 1000
num_samples = 48000
sampling_rate = 48000
amplitude = 16000
file = "Audio_Sin.wav"

#Phương trình sóng sin_wave
sine_wave = [np.sin(2*np.pi*frequency*x/sampling_rate) for x in
range(num_samples)]

#Thiết lập các khung frames hoặc mẫu
nframes = num_samples
comptype = "NONE"
compname = "not compressed"
nchannels = 1
sampwidth = 2      #16bit

#Mở file
wav_file = wave.open(file, 'w')
wav_file.setparams((nchannels, sampwidth, int(sampling_rate), nframes, comptype,
compname))

#Thiết lập thông số cho file
for s in sine_wave:
    wav_file.writeframes(struct.pack('h', int(s*amplitude)))

#Đóng gói file
wav_file.close()
```



Câu 3:

a)A/C trình bày các mô hình màu RGB, YIQ, YUV và YCrCb.



- **4:4:4 (No subsampling):** Trong mô hình này, mỗi kênh màu (ví dụ: RGB - đỏ, xanh lá, lam) đều được lưu trữ một cách đầy đủ cho mỗi pixel. Điều này có nghĩa là mọi pixel đều có thông tin về tất cả các kênh màu, không có việc giảm bớt thông tin màu sắc.
- **4:2:2:** Ở đây, mẫu màu sắc được giảm xuống còn 50% theo chiều ngang (horizontal) so với 4:4:4. Cụ thể, thông tin về màu sắc chỉ được lưu trữ cho mỗi cặp pixel theo chiều ngang, trong khi các pixel cùng hàng dọc vẫn chia sẻ cùng thông tin về màu sắc.
- **4:1:1:** Trong mô hình này, mẫu màu sắc bị giảm xuống 75% theo chiều ngang và chỉ 25% theo chiều dọc so với 4:4:4. Thông tin về màu sắc chỉ được lưu trữ cho mỗi bốn pixel.
- **4:2:0:** Đây là mô hình subsampling phổ biến trong video và ảnh số. Màu sắc được giảm xuống 50% theo cả chiều ngang và chiều dọc so với 4:4:4. Trong khi thông tin về màu sắc của mỗi hàng vẫn được lưu trữ, thông tin về màu sắc của mỗi hai hàng liên tiếp chỉ được lưu trữ cho một hàng

- **Mô hình màu RGB**: mỗi màu hiển thị được mô tả gồm các tham số độc lập - độ chói của các tham số chính được sử dụng trong màn hình CRT màu. Là mô hình màu được kết hợp bởi 3 thành phần cơ bản là Red – Green – Blue (RGB) để tạo ra các màu sắc khác nhau
- **Mô hình màu YIQ** (Luminance, In-phase, Quadrature): là mô hình màu được sử dụng trong truyền hình analog của Mỹ, đặc biệt là trong hệ thống truyền hình NTSC. Y đại diện cho độ sáng (luminance) của hình ảnh (thành phần này tương đương với đen trắng trong hệ thống truyền hình cũ).I và Q là hai thành phần màu chịu trách nhiệm truyền tải thông tin về sắc độ (chrominance), với I (In-phase): Thành phần chênh lệch pha. Q (Quadrature): Thành phần vuông pha.
- **Mô hình màu YUV** (Luminance, Chrominance): là mô hình màu được sử dụng trong truyền hình màu, đặc biệt là hệ thống PAL và SECAM ở châu Âu và một số nơi khác. Y đại diện cho độ sáng (luminance), tương tự như trong YIQ. U và V là hai thành phần màu sắc (chrominance) xác định sự khác biệt giữa màu xanh lam và độ sáng (U) cũng như sự khác biệt giữa màu đỏ và độ sáng (V).
- **Mô hình màu YCrCb** (Luminance, Chrominance Blue, Chrominance Red) là phiên bản số hóa của YUV, được sử dụng chủ yếu trong các hệ thống xử lý video kỹ thuật số, bao gồm nén hình ảnh (JPEG) và nén video (MPEG). Y đại diện cho độ sáng (luminance). Cr (Chrominance Red) đại diện cho sự khác biệt giữa màu đỏ và độ sáng. Cb (Chrominance Blue) đại diện cho sự khác biệt giữa màu xanh lam và độ sáng.

b)Dữ liệu video có tốc độ truyền 24 fps, mỗi frame có kích thước 720x486, mô hình YCrCb chroma subsampling sử dụng là 4:2:2.Xác định dung lượng lưu trữ video trong 1 phút 15 giây.

Vì là mô hình YcrCb chroma subsampling sử dụng là 4:2:2 nên mỗi điểm ảnh pixel của ảnh sẽ dùng 2 bytes.

Vậy nên:

Tốc độ truyền Bit-rate = H x W x Dept x FPS = 720 x 486 x 2 bytes x 24 = 16 796 160 (bytes/s)

Dung lượng lưu trữ video trong 1p15s (75 giây):

C = Bit-rate x thời gian = 16796160 x 75 = 1 259 712 000 bytes

Câu 4:

a)Viết chương trình hiển thị nội dung ảnh có định dạng jpg, png, bitmap

```
#Câu 4
!pip install Pillow
#a) Chương trình hiển thị nội dung ảnh có định dạng JPG, PNG, Bitmap
from PIL import Image
def display_image(image_path):
    # Mở ảnh
    image = Image.open(image_path)
    # Hiển thị ảnh
    image.show()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/image/sample-boat-400x300.png"  # Thay đổi đường
dẫn đến ảnh của bạn
display_image(image_path)
```

b)Viết chương trình thực hiện các chức năng biến đổi ảnh như : làm mờ ảnh

```
#b) Chương trình làm mờ ảnh
from PIL import Image, ImageFilter
def blur_image(image_path, output_path):
    # Mở ảnh
    image = Image.open(image_path)
    # Làm mờ ảnh
    blurred_image = image.filter(ImageFilter.BLUR)
    # Lưu ảnh làm mờ
    blurred_image.save(output_path)
    # Hiển thị ảnh làm mờ
    blurred_image.show()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/image/sample-boat-400x300.png"
output_path = "blurred_image.png"
blur_image(image_path, output_path)
```

c) Viết chương trình thực hiện các chức năng biến đổi ảnh như xoay ảnh theo 1 góc tùy ý (góc <=20 độ)

```
#c) Chương trình xoay ảnh theo một góc tùy ý (góc <= 20 độ)
from PIL import Image
def rotate_image(image_path, output_path, angle):
    # Mở ảnh
    image = Image.open(image_path)
    # Xoay ảnh theo góc
    rotated_image = image.rotate(angle)
    # Lưu ảnh sau khi xoay
    rotated_image.save(output_path)
    # Hiển thị ảnh đã xoay
    rotated_image.show()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/image/sample-boat-400x300.png"
output_path = "rotated_image.png"
angle = 20  # Xoay 20 độ
rotate_image(image_path, output_path, angle)
```

d) Viết chương trình thực hiện các chức năng biến đổi ảnh như co giãn ảnh.

```
#d) Chương trình co giãn ảnh
from PIL import Image
def resize_image(image_path, output_path, new_width, new_height):
    # Mở ảnh
    image = Image.open(image_path)
    # Co giãn ảnh
    resized_image = image.resize((new_width, new_height))
    # Lưu ảnh sau khi co giãn
    resized_image.save(output_path)
    # Hiển thị ảnh sau khi co giãn
    resized_image.show()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/image/sample-boat-400x300.png"
output_path = "resized_image.png"
new_width = 400
new_height = 300
resize_image(image_path, output_path, new_width, new_height)
```

e) Viết chương trình thực hiện các chức năng biến đổi ảnh như lật ảnh từ trái sang phải(hoặc ngược lại)

```
#e) Chương trình lật ảnh từ trái sang phải (hoặc ngược lại)
from PIL import Image
def flip_image_left_right(image_path, output_path):
    # Mở ảnh
    image = Image.open(image_path)
    # Lật ảnh trái sang phải
    flipped_image = image.transpose(Image.FLIP_LEFT_RIGHT)
    # Lưu ảnh đã lật
    flipped_image.save(output_path)
    # Hiển thị ảnh sau khi lật
    flipped_image.show()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/image/sample-boat-400x300.png"
output_path = "flipped_image_lr.png"
flip_image_left_right(image_path, output_path)
```

f) Viết chương trình thực hiện các chức năng biến đổi ảnh như lật ảnh từ trên xuống dưới(hoặc ngược lại).

```
#f) Chương trình lật ảnh từ trên xuống dưới (hoặc ngược lại)
from PIL import Image
def flip_image_top_bottom(image_path, output_path):
    # Mở ảnh
    image = Image.open(image_path)
    # Lật ảnh từ trên xuống dưới
    flipped_image = image.transpose(Image.FLIP_TOP_BOTTOM)
    # Lưu ảnh đã lật
    flipped_image.save(output_path)
    # Hiển thị ảnh sau khi lật
    flipped_image.show()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/image/sample-boat-400x300.png"
output_path = "flipped_image_tb.jpg"
flip_image_top_bottom(image_path, output_path)
```

Câu 5: Viết chương trình phát hiện khuôn mặt người trên ảnh.

```
#5Chương trình phát hiện khuôn mặt
!pip install opencv-python
import cv2
def detect_faces(image_path, output_path):
    # Tải mô hình phát hiện khuôn mặt
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

    # Đọc ảnh
    image = cv2.imread(image_path)
    # Chuyển ảnh sang ảnh xám
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Phát hiện khuôn mặt
    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))

    # Vẽ hình chữ nhật quanh các khuôn mặt phát hiện được
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Lưu ảnh đã xử lý
    cv2.imwrite(output_path, image)

    # Hiển thị ảnh đã xử lý
    cv2.imshow("Faces detected", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
# Ví dụ sử dụng
image_path = "D:/Multimedia/Data/human.jpg"
output_path = "detected_faces.jpg"  # Đường dẫn lưu ảnh đã xử lý
detect_faces(image_path, output_path)
```