**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY**

**DEPARTMENT OF TELECOMMUNICATIONS ENGINEERING**

**--------------------------**



**COMPUTING SYSTERMS AND PROGRAMMING  
C - PROJECT**

**Lecturer : Assoc. Prof. Đặng Thành Tín**

**Students : Đinh Hoàng Luân \_ 2051145**

*HCM City 2022*

**TOPICS**

**PROJECT: C programming**

This takes you 10% of overall score on my side

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

The requirement for the project:  
 1. Use functions as much as possible  
 2. Create user interface as clear and beautiful as possible  
 3. Check range for every value input and output appropriately  
 4. The program is organized well for structure programming   
 5. The program needs to comment as much as possible  
 6. Each student must choose only one topic  
 7. The file to submit to BKel right after the final exam includes 1 file of .cpp.

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

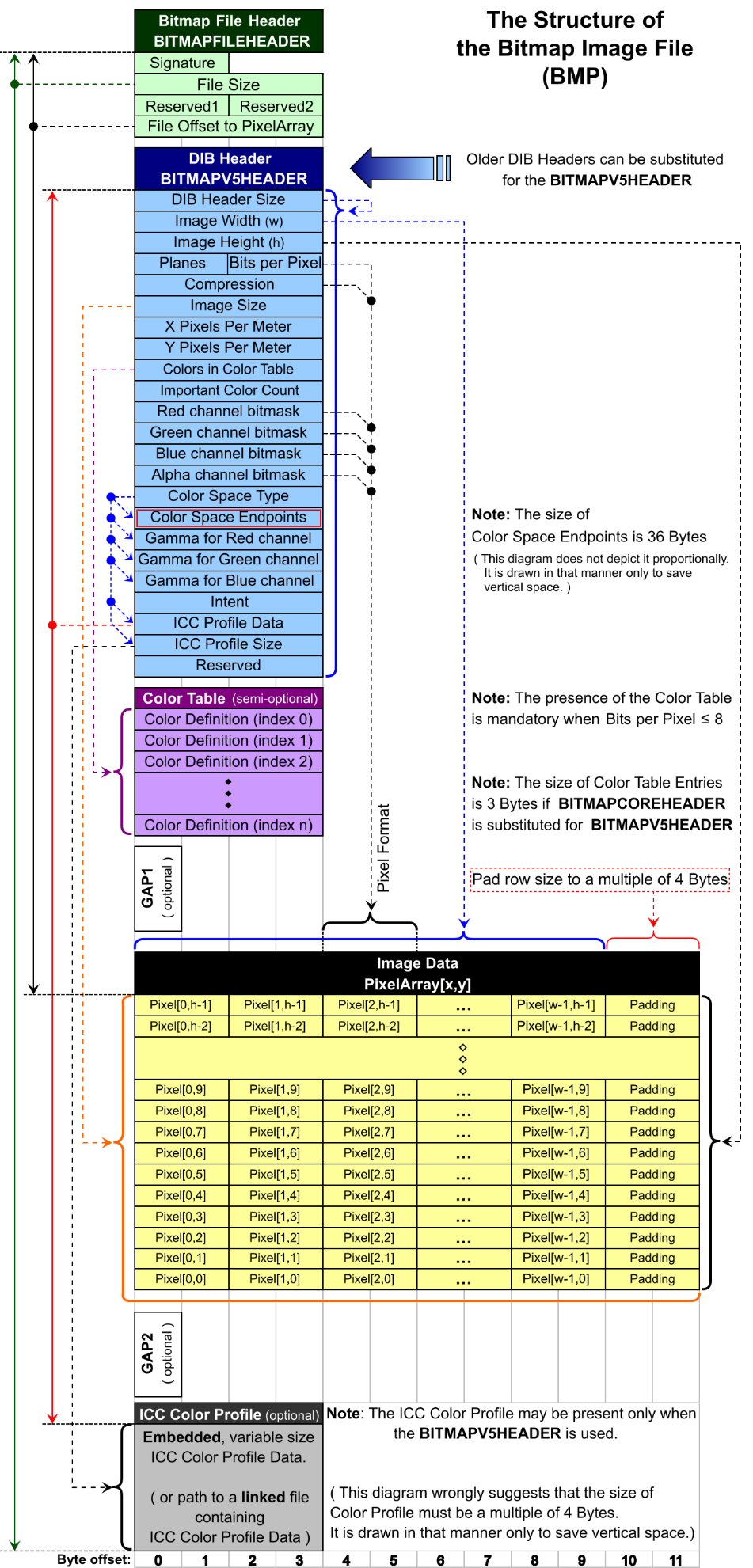
5. Write the program to convert the image file .BMP of 256 grey levels to binary image .BMP for the data section in the image file with the rule:

gray levels binary

0 – 127 0  
 128 – 255 1

1. **Definition** **Bitmap file format**- The **BMP file format**, also known as **bitmap image file**, **device independent bitmap (DIB) file format** and **bitmap**, is a [raster graphics](https://en.wikipedia.org/wiki/Raster_graphics) [image file format](https://en.wikipedia.org/wiki/Image_file_format) used to store [bitmap](https://en.wikipedia.org/wiki/Bitmap) [digital images](https://en.wikipedia.org/wiki/Digital_image), independently of the [display device](https://en.wikipedia.org/wiki/Display_device) (such as a [graphics adapter](https://en.wikipedia.org/wiki/Graphics_adapter)), especially on [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) and [OS/2](https://en.wikipedia.org/wiki/OS/2) operating systems.  
   - The BMP file format is capable of storing [two-dimensional](https://en.wikipedia.org/wiki/2D_computer_graphics) digital images both [monochrome](https://en.wikipedia.org/wiki/Monochrome) and color, in various [color depths](https://en.wikipedia.org/wiki/Color_depth), and optionally with [data compression](https://en.wikipedia.org/wiki/Data_compression), [alpha channels](https://en.wikipedia.org/wiki/Alpha_compositing), and [color profiles](https://en.wikipedia.org/wiki/Color_management). The [Windows Metafile](https://en.wikipedia.org/wiki/Windows_Metafile) (WMF) specification covers the BMP file format.
   1. ***File structure*-** The bitmap image file consists of fixed-size structures (headers) as well as variable-sized structures appearing in a predetermined sequence. Many different versions of some of these structures can appear in the file, due to the long evolution of this file format.

|  |  |  |  |
| --- | --- | --- | --- |
| **Structure name** | **Size** | **Purpose** | **Comments** |
| Bitmap file header | 14 bytes | To store general information about the bitmap image file | Not needed after the file is loaded in memory |
| DIB header | Fixed-size (7 different versions exist) | To store detailed information about the bitmap image and define the pixel format | Immediately follows the Bitmap file header |
| Pixel array | Variable size | To define the actual values of the pixels | The pixel format is defined by the DIB header or Extra bit masks. Each row in the Pixel array is padded to a multiple of 4 bytes in size |



### *Bitmap file header* - This block of bytes is at the start of the file and is used to identify the file. A typical application reads this block first to ensure that the file is actually a BMP file and that it is not damaged. The first 2 bytes of the BMP file format are the character "B" then the character "M" in [ASCII](https://en.wikipedia.org/wiki/ASCII) encoding. All of the integer values are stored in [little-endian](https://en.wikipedia.org/wiki/Little-endian) format.

|  |  |  |
| --- | --- | --- |
| **Offset hex** | **Size** | **Purpose** |
| 00 | 2 bytes | Identifier (usually BM) |
| 02 | 4 bytes | The size of the BMP file in bytes |
| 06 | 2 bytes | Reserved; actual value depends on the application that creates the image, if created manually can be 0 |
| 08 | 2 bytes | Reserved; actual value depends on the application that creates the image, if created manually can be 0 |
| 0A | 4 bytes | Offset to pixel array |

* 1. ***DIB header (bitmap information header)***- This block of bytes tells the application detailed information about the image, which will be used to display the image on the screen. The block also matches the header used internally by Windows and OS/2 and has several different variants. All of them contain a dword (32-bit) field, specifying their size, so that an application can easily determine which header is used in the image.

|  |  |  |
| --- | --- | --- |
| **Offset hex** | **Size** | **Purpose** |
| 0E | 4 bytes | DIB header size |
| 12 | 4 bytes | Image width |
| 16 | 4 bytes | Image height |
| 1C | 2 bytes | Bit per Pixel |

* 1. ***Pixel array***- First pixel that located in the pixel array is the left-bottom pixel of the image.  
     - Pixels are stored "bottom-up", starting in the lower left corner, going from left to right, and then row by row from the bottom to the top of the image  
     - Bytes in a row in the pixel array must be multiple of four bytes with  
      **Rowsize**
  2. ***Pixel format***- The 24-bit per pixel (24bpp) format supports 16,777,216 distinct colors and stores 1 pixel value per 3 bytes. Each pixel value defines the red, green and blue samples of the pixel (8.8.8.0.0 in RGBAX notation). Specifically, in the order: blue, green and red (8 bits per each sample).  
     - The 1-bit per pixel (1bpp) format supports 2 distinct colors, (for example: black and white). The pixel values are stored in each bit, with the first (left-most) pixel in the most-significant bit of the first byte. Each bit is an index into a table of 2 colors.- The 8-bit per pixel (8bpp) format supports 256 distinct colors and stores 1 pixel per 1 byte. Each byte is an index into a table of up to 256 colors.

1. **Project's algorithm**

* *I divided my project into* ***3 main*** *parts:*
* **Part 1:** I check the picture is bitmap by read the 2 first bit.
* **Part 2:** I continue check the width, height, and the byte offset.
* **Part 3:** Check the Bit per Pixel, and print out
* **Part 4:** Main code

***Part 1: Check the picture is bitmap by read the 2 first bit.***

/\* Confirm the picture have already converted into bmp or not \*/

unsigned char getByte(FILE \*fp, int location){

unsigned char byte;

fseek(fp,location,SEEK\_SET);

fread(&byte,1,1,fp);

return byte;

}

***Part 2: Check the width, height, and the byte offset.***

/\* Move to address 0x12 to take the bitmap width in pixels (signed integer) \*/

int getwidth(FILE \*fp){

int width;

fseek(fp,0x12,SEEK\_SET);

fread(&width,1,4,fp);

return width;

}

/\* Move to address 0x16 to take the bitmap height in pixels (signed integer) \*/

int getheight(FILE \*fp){

int height;

fseek(fp,0x16,SEEK\_SET);

fread(&height,1,4,fp);

return height;

}

/\* Move to address 0xA to take the offset, starting address of the byte where the bitmap image data (pixel array) can be found. \*/

int getOffsetToPixelArray(FILE \*fp){

int OffsetToPixelArray;

fseek(fp,0xA,SEEK\_SET);

fread(&OffsetToPixelArray,1,4,fp);

return OffsetToPixelArray;

}

***Part 3: Check Bit per Pixel and print out.***

/\* Move to address 0x1C to take the number of bits per pixel, which is the color depth of the image. Typical values are 1, 4, 8, 16, 24 and 32 \*/

int getBitPerPixel(FILE \*fp){

int BitPerPixel;

fseek(fp,0x1C,SEEK\_SET);

fread(&BitPerPixel,1,1,fp);

return BitPerPixel;

}

/\* Print "0" if getByte smaller or equal 127, and print "1" if getByte larger 127 \*/

void printImage(FILE \*fp, int height, int width, int OffsetToPixelArray, int BitPerPixel){

int RowSize= int(((BitPerPixel\*width)+31)/32)\*4;

for(int i= height -1; i>=0; i--){

for(int j=0; j< width; j++ ){

if((getByte(fp, OffsetToPixelArray + i\*RowSize + j)) <= 127)

printf("0");

else

printf("1");

}

printf("\n");

}

}

***Part 4: Main code***

int main(){

int a;

printf("#########################################################\n");

printf("#### HELLO PROFESSOR ####\n");

printf("#### ####\n");

printf("#### My name is DINH HOANG LUAN ####\n");

printf("#### ####\n");

printf("#### From TT03 ####\n");

printf("#### ####\n");

printf("#### Today, I will present for you topic 5 ####\n");

printf("#### ####\n");

printf("#### ####\n"); printf("#########################################################\n");

/\* Declare fp \*/

FILE \*fp=fopen("new.bmp","r");

/\* Declare height, width, offset \*/

int height = getheight(fp);

int width = getwidth(fp);

int OffsetToPixelArray = getOffsetToPixelArray(fp);

int b;

printf("\n####################################################\n");

printf("Press buttom 0 to check the picture is the bitmap or not");

printf("######################################################\n");

printf("Please select buttom 0: ");

scanf("%d",&b);

if( b == 0){

/\* If the picture is bitmap type then print out \*/

if((getByte(fp,0) != 'B') || (getByte(fp,1) != 'M'))

printf("It is not a bmp!!. Please take another the picture.");

else

printf("This is a bitmap picture: %c%c\n", getByte(fp,0), getByte(fp,1));

}

int c;

printf("\n####################################################\n");

printf("## Press buttom 1 to see the feature of the picture ##\n");

printf("######################################################\n");

printf(" Please select buttom 1: ");

scanf("%d",&c);

if(c == 1){

/\* Print the width of picture \*/

printf(" The width is %d", getwidth(fp));

/\* Print the height of picture \*/

printf("\n The height is %d", getheight(fp));

/\* Print the offset of picture \*/

printf("\n The byte offset is %d", getOffsetToPixelArray(fp));

}

int d;

printf("\n########################################################\n");

printf("#### Press buttom 2 to check the Bit per Pixel ####\n");

printf("##########################################################\n");

printf(" Please select buttom 2: ");

scanf("%d",&d);

if(d == 2){

/\* If bit per pixel euqal 8 then it legal \*/

if(getBitPerPixel(fp) != 8)

printf(" Invalid picture. Stop");

else

printf(" Bit per Pixel is %d\n",getBitPerPixel(fp));

}

int e;

printf("\n########################################################\n");

printf("##### Press buttom 3 to print the picture ####\n");

printf("##########################################################\n");

printf(" Please select buttom 3: ");

scanf("%d", &e);

if (e == 3){

int BitPerPixel = getBitPerPixel(fp);

printImage(fp, height, width, OffsetToPixelArray, BitPerPixel);

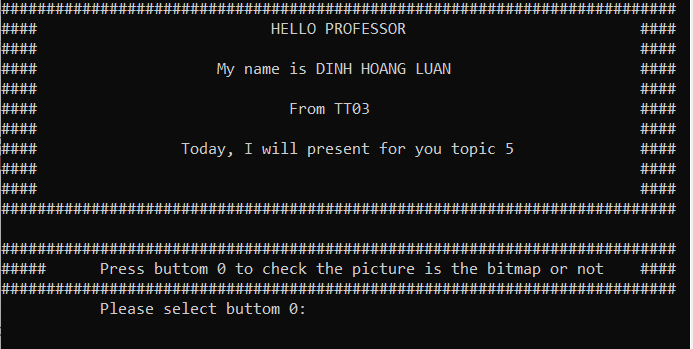
}

return 0;

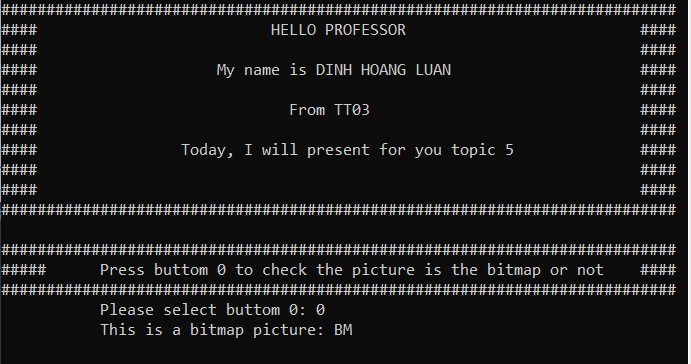
}

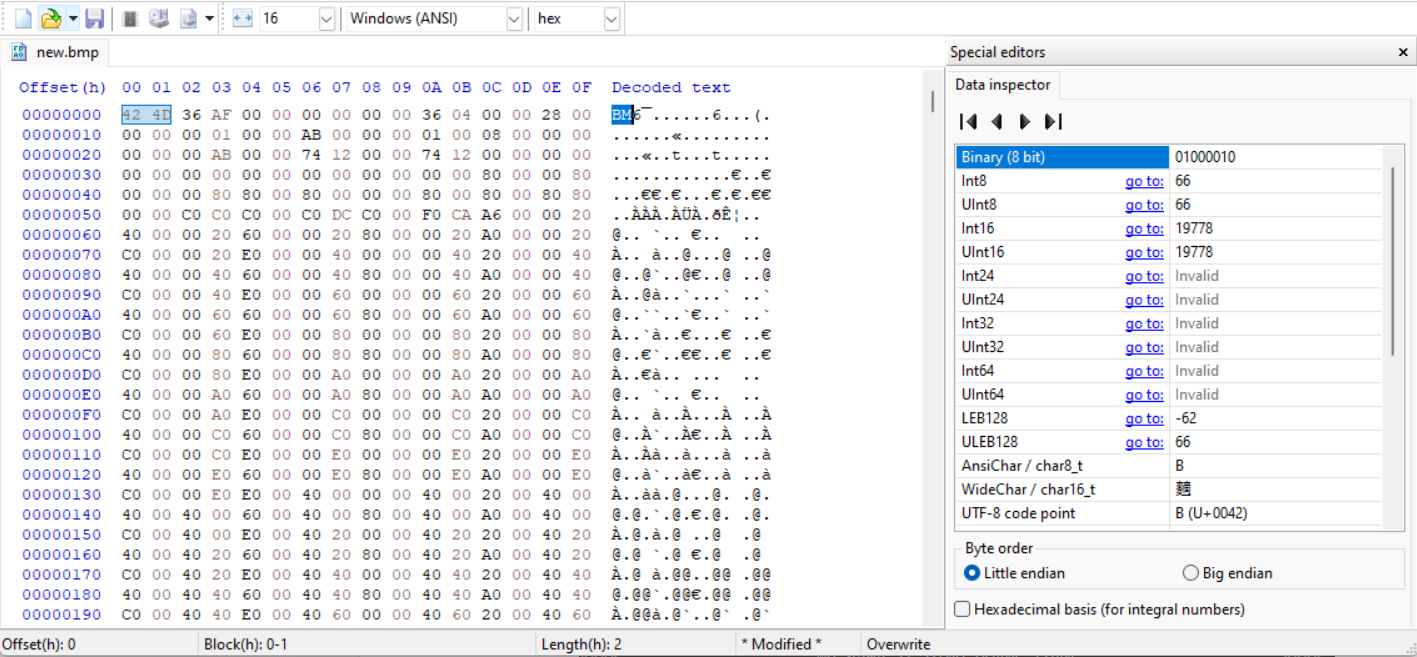
1. **Demontration**

***3.1) Check the picture is bitmap or not***

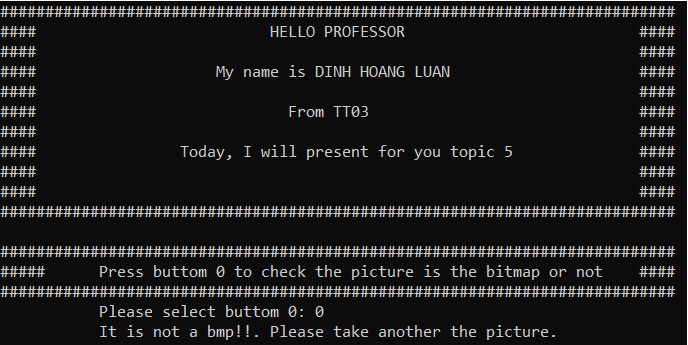


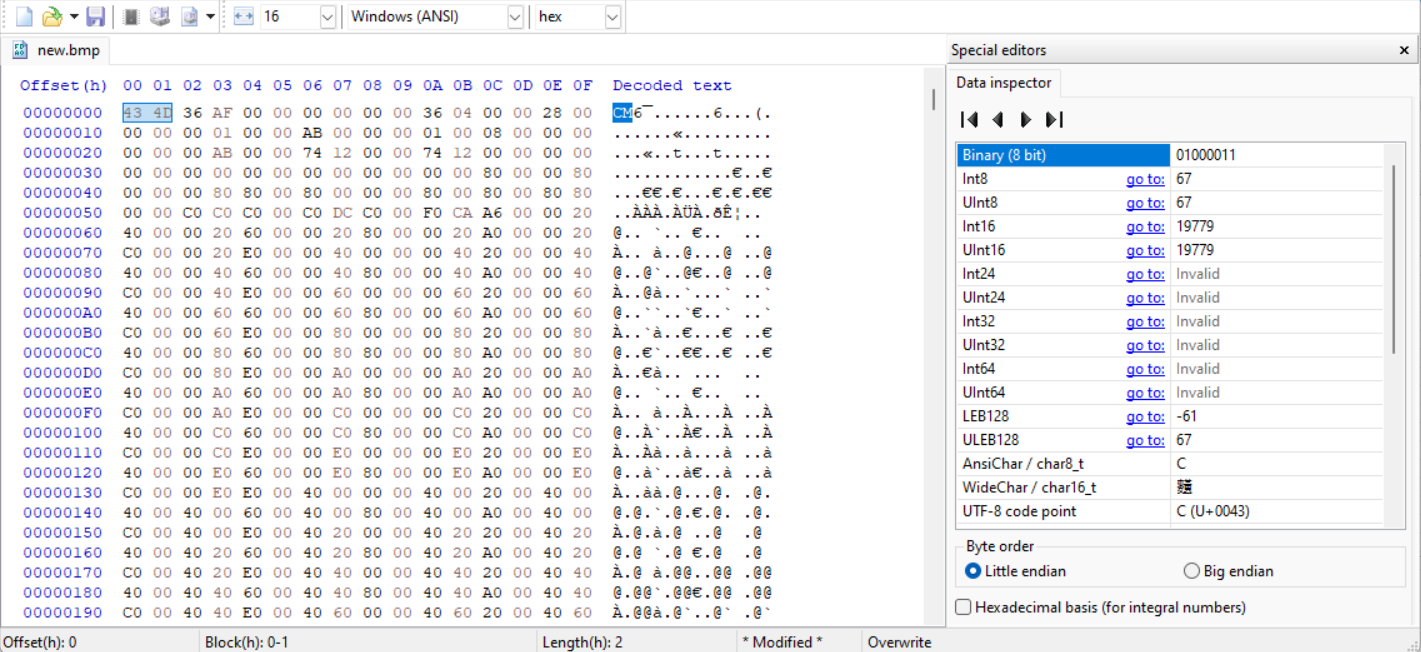
* The picture below is bitmap file.



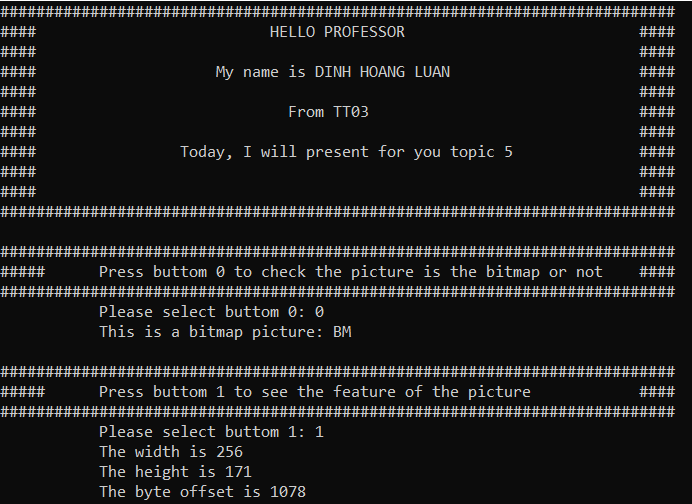


* The picture below is not bitmap file, find the legal picture.

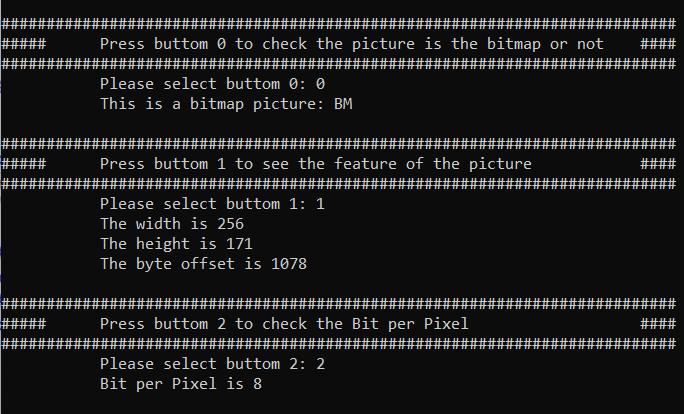




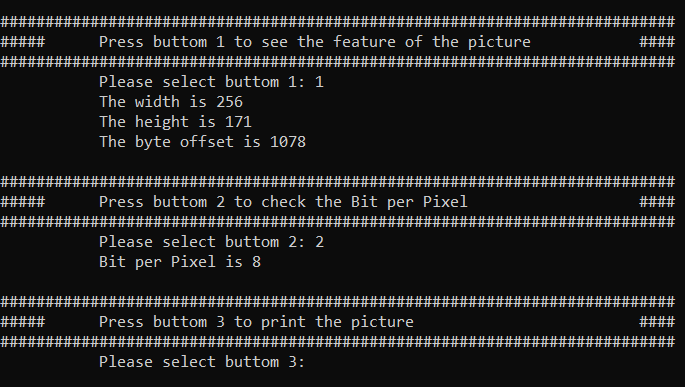
***3.2) Check the width, height, and the byte offset.***



***3.3) Check Bit per Pixel***

******

***3.4) Result***

******

******

1. **Operate project**

* Step 1: You press the button “0” to check picture
* Step 2: You press the button “1” to check the features (width, height, offset)
* Step 3: You press the button “2” to check the Bit per Pixel
* Step 4: You press the button “3” to print out the picture, in order to see clearly the picture, you should hold the **“ctrl + scroll button”** then zoom it with the maximum.

