HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

OFFICE FOR INTERNATIONAL STUDY PROGRAM

FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING

——————– \* ———————

Logo, company name

Description automatically generated

COMPUTER SYSTEM AND PROGRAMMING

C PROJECT REPORT

**IMAGE TO BINARY BITMAP CONVERTER**

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

**Class:** TT03

**Semester:** 212

**Members:** Lương Triển Thắng

**Student ID:** 2051194

*Hồ Chí Minh City – 15th May, 2022*

Table of Contents

[I. Requiremnts 4](#_Toc103453642)

[II. Features 4](#_Toc103453643)

[III. Instructions 4](#_Toc103453644)

[IV. Images 5](#_Toc103453645)

[V. Demonstration 6](#_Toc103453646)

[VI. Understanding bitmap file format 7](#_Toc103453647)

[1. File structure 7](#_Toc103453648)

[2. Bitmap and DIB header 7](#_Toc103453649)

[3. Pixel format 7](#_Toc103453650)

[4. Pixel array 8](#_Toc103453651)

[VII. Algorithm 8](#_Toc103453652)

[1. Turning BGR to grayscale 8](#_Toc103453653)

[2. Getting color bytes 8](#_Toc103453654)

[VIII. Functions 8](#_Toc103453655)

[1. Image processing 8](#_Toc103453656)

[a. Read a byte/bytes 8](#_Toc103453657)

[b. Check whether a bitmap file or not 9](#_Toc103453658)

[c. Get width, get height, get bpp, get offset to image array 9](#_Toc103453659)

[d. Turn an array of 8 binary integers into a character (a byte) 9](#_Toc103453660)

[e. Integer to four bytes (characters) 9](#_Toc103453661)

[f. Printing monochrome image 10](#_Toc103453662)

[g. Convert the image into monochrome array 10](#_Toc103453663)

[h. Make a bitmap binary image file array 12](#_Toc103453664)

[i. Write a new binary bitmap image 13](#_Toc103453665)

[2. UI related (Windows only) 13](#_Toc103453666)

[a. char to wchar 14](#_Toc103453667)

[b. Get screen resolution 14](#_Toc103453668)

[c. Get current font width 14](#_Toc103453669)

[d. Set console window and font size 14](#_Toc103453670)

[e. Set print size 15](#_Toc103453671)

[f. Set default console size 15](#_Toc103453672)

[IX. Complete code 16](#_Toc103453673)

[1. Main 16](#_Toc103453674)

[2. bmpLib 20](#_Toc103453675)

[a. bmpLib.h 20](#_Toc103453676)

[b. bmpLib.cpp 20](#_Toc103453677)

[3. bmpWrite 24](#_Toc103453678)

[a. bmpWrite.h 24](#_Toc103453679)

[b. bmpWrite.cpp 25](#_Toc103453680)

[4. UILib 27](#_Toc103453681)

[a. UILib.h 27](#_Toc103453682)

[b. UILib.cpp 27](#_Toc103453683)

[X. Appendix 30](#_Toc103453684)

[XI. References 30](#_Toc103453685)

# Requiremnts

* Use subroutines as much as possible.
* Create user interface as clear and beautiful as possible.
* Check range for every value input and output appropriately.
* The program should be organized so well for structure programming.
* The program needs to comment as mamy as possible.  
  The detail explanation of the program will be describe more here.
* Convert the image file .BMP of 256 grey levels to binary image .BMP.

# Features

* Friendly UI.
* Drag and drop file input.
* Currently support 8bpp and 24bpp bitmap image.
* Builtin help screen.
* Print the image to the console window.
* Convert the image into binary (monochrome) bitmap image (1bpp) with progress displayed.

# Instructions

* User will be greeting with a welcome screen.
* If you’re first using the program, type “h” or “help” to show the instructions.
* User inputs a bitmap image file path with drag and drop supported.
* Type “p” for print or “c” to convert.
* Print:
* Console and font will be resized for better viewing.
* Press <Enter> once, then please maximize the window by clicking on the middle icon at the top right corner.
* Press <Enter> again to print monochrome version of the image.
* Press <Enter> after printing to restore console size and font.
* Convert:
* The image will be converted to monochrome bitmap file and saved at the same location as the input.

# Images

A computer screen capture

Description automatically generated with low confidence

Figure 1 - Welcome Screen

Text

Description automatically generated

Figure 2 – Instructions

A picture containing text, window, curtain, door

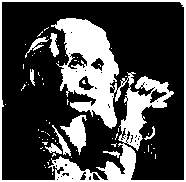
Description automatically generated 

Figure 3 - Image printed in console and converted bitmap image

# Demonstration

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| 8bpp grayscale, 186x182 px |  |  |
| 8bpp grayscale, 512x512 px |  |  |
| 8bpp colored, 512x384 px |  |  |
| 24bpp, 1920x1080 |  |  |

# Understanding 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 image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter), especially on Microsoft Windows and OS/2 operating systems. [1]

## File structure

There are 8 parts that construct a bitmap file. But there are only 3 that we need to consider. (… are unimportant parts)

|  |  |  |
| --- | --- | --- |
| Structure name | Size | Purpose |
| Bitmap file header | Fixed - 14 bytes | Store general information about the bitmap image file. |
| DIB file header | Fixed – 62 bytes | Store detailed information about the bitmap image and define the pixel format. |
| … |  |  |
| Pixel array | Variable – depends on image bpp and size | Define the actual values of the pixels. |
| … |  |  |

## Bitmap and DIB header

|  |  |  |  |
| --- | --- | --- | --- |
|  | Offset | Size | Purpose |
| Bitmap file header | 00 | 2 bytes | Identifier (usually BM) |
| 02 | 4 bytes | Size of bmp file |
| 06 | 4 bytes | Reserved |
| 0A | 4 bytes | Offset to pixel array |
| DIB file header | 0E | 4 bytes | DIB header size |
| 12 | 4 bytes | Image width |
| 16 | 4 bytes | Image height |
| … |  |  |
| 1C | 2 bytes | Bit per pixel |
| … |  |  |
| 36 | 4 bytes | Start of color table first color table entry (for 1bpp) |
| 40 | 4 bytes | last color table entry (for 1bpp) |
| … |  |  |

## Pixel format

* 1bpp: Image only consists of two colors, which predefined at offset 36 and 40. Each bit defines one pixel of the image.
* 8bpp: Image supports 256 colors. Each byte is an index into color table.
* 24bpp: Image supports colors. Each pixel defined by 3 bytes, each byte is the strength of blue, green, red.
* 2bpp, 4bpp, 16 bpp, 32bpp: Learn more at [1]

## Pixel array

* First pixel that defines in the pixel array is the left-bottom pixel of the image.
* So 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. Thus, each row must consists of .
* So padding bytes must be appended to the end of the rows in order to bring up the length of the rows to a multiple of four bytes. They can be any values.

# Algorithm

## Turning BGR to grayscale [2]

* For 8bpp image: each byte represents an index to color table. Each color in color table is 4 bytes (blue, green, red, alpha)  
  For 24bpp image: 3 bytes of a pixel are the strengths of blue, green, red
* To turn BGR to grayscale color, apply this formula (luminosity method)

## Getting color bytes

* For 8bpp image: location of pixel is

With blue value at , green at and red at .

* For 24bpp image: location of pixel is

With blue value at , green at and red at .

# Functions

## Image processing

* Libraries: stdio, iostream, malloc

### Read a byte/bytes [3]

* Purpose: get the value in specific location.
* Input: file pointer, location to byte, number of bytes
* Output: the value (char or int)

uchar getByte**(**FILE **\***fp**,** **int** location**)** **{**

uchar byte**;**

fseek**(**fp**,**location**,**SEEK\_SET**);**

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

**return** byte**;**

**}**

uchar getBytes**(**FILE **\***fp**,** **int** location**,** **int** numberOfBytes**)** **{**

uchar byte**;**

fseek**(**fp**,**location**,**SEEK\_SET**);**

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

**return** byte**;**

**}**

### Check whether a bitmap file or not [3]

* Read first two bytes if equals to or not.
* Input: file pointer
* Output: 0 or 1

**int** IsBitMap**(**FILE **\***fp**)** **{**

ushort s**;**

fread**(&**s**,**1**,**2**,**fp**);**

**return** s**==**19778 **?** 1 **:** 0**;**

**}**

### Get width, get height, get bpp, get offset to image array [3]

*//Get the width of the picture, in 18-21 bytes*

**int** getWidth**(**FILE **\***fp**)** **{**

**int** width**;**

fseek**(**fp**,**18**,**SEEK\_SET**);**

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

**return** width**;**

**}**

*//Get the height of the picture, in 22-25 bytes*

**int** getHeight**(**FILE **\***fp**)** **{**

**int** height**;**

fseek**(**fp**,**22**,**SEEK\_SET**);**

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

**return** height**;**

**}**

*//Get the number of bits of each pixel in 28-29 bytes*

ushort getBit**(**FILE **\***fp**)** **{**

ushort bit**;**

fseek**(**fp**,**28**,**SEEK\_SET**);**

fread**(&**bit**,**1**,**2**,**fp**);**

**return** bit**;**

**}**

*//Get the starting position of data, in 10-13 bytes*

uint getOffSet**(**FILE **\***fp**)** **{**

uint OffSet**;**

fseek**(**fp**,**10L**,**SEEK\_SET**);**

fread**(&**OffSet**,**1**,**4**,**fp**);**

**return** OffSet**;**

**}**

### Turn an array of 8 binary integers into a character (a byte)

* Example: {0,1,1,1,0,0,0,1} 🡪 71
* Input: binary array
* Output: a byte

**char** BitToByte**(int** **\***num**)** **{**

**char** result **=** 0**;**

**for** **(int** i **=** 0**;** i **<** 8**;** **++**i **)**

result **|=** **(**num**[**i**]** **==** 1**)** **<<** **(**7 **-** i**);**

**return** result**;**

**}**

### Integer to four bytes (characters)

* Example: 1078 🡪 36 04 00 00
* Input: an integer, 4-bytes array pointer
* Output: 4-bytes array

**void** ToFourBytes**(unsigned** **long** n**,** **char** bytes**[**4**]){**

bytes**[**0**]** **=** **(**n **>>** 24**)** **&** 0xFF**;**

bytes**[**1**]** **=** **(**n **>>** 16**)** **&** 0xFF**;**

bytes**[**2**]** **=** **(**n **>>** 8**)** **&** 0xFF**;**

bytes**[**3**]** **=** **(**n **>>** 0**)** **&** 0xFF**;**

**}**

### Printing monochrome image

* Use the algorithm from VII.2 to get location of pixels and turn pixel data into grayscale using VII.1.
* Since the pixels are stored "bottom-up", we will print from the last row up to the top.
* Input: file pointer, width, height, bpp

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

uint offset **=** getOffSet**(**fp**);**

**int** rowSize **=** **int((**bpp**\***width**\***1.0 **+** 31**)/**32**)\***4**;**

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

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

**float** gray **=** 0**;**

uchar red **=** 0**,** green **=** 0**,** blue **=** 0**;**

**if(**bpp **==** 8**){**

uchar index **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**);**

uint location **=** 54 **+** index **\*** 4**;**

blue **=** getByte**(**fp**,** location**);**

green **=** getByte**(**fp**,** location **+** 1**);**

red **=** getByte**(**fp**,** location **+** 2**);**

**}** **else** **if(**bpp **==** 24**)** **{**

blue **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3**);**

green **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**\***3 **+** 1**);**

red **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3 **+** 2**);**

**}**

gray **=** 0.3**\***red **+** 0.59**\***green **+** 0.11**\***blue**;**

**if(**gray **<=** 127**)**

printf**("0");**

**else**

printf**("1");**

**}**

printf**("\n");**

**}**

**}**

### Convert the image into monochrome array

* Use the algorithm from VII.2 to get location of pixels and turn pixel data into grayscale using VII.1.
* In this function, we will convert from top for ease.
* A new array with size is created to store binary value of each pixel.
* After the conversion, every 8 bits in the array will be convert into a byte using BitToByte function which described at section d.
* Input: file pointer, width, height, bpp, data size pointer
* Output: monochrome data array (8 bits), data size

**char** **\***GetMonochromeData**(**FILE **\***fp**,** **int** width**,** **int** height**,** **int** bpp**,** **unsigned** **long** **\***dataSize**)** **{**

uint offset **=** getOffSet**(**fp**);**

**int** rowSize **=** **int((**bpp**\***width**\***1.0 **+** 31**)/**32**)\***4**;**

**unsigned** **long** newRowSizeInBits **=** **int((**1**\***width**\***1.0 **+** 31**)/**32**)\***4**\***8**;**

**char** **\***monochromeBits **=** **(char** **\*)**calloc**(**newRowSizeInBits **\*** height**,** **sizeof(char));**

**int** progressCounter **=** 5**;**

system**("cls");**

printf**("Converting... Please wait... (Ctrl+C to break)\nProgress: 0 %%");**

**for(int** i **=** 0**;** i **<** height**;** i**++)** **{**

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

**float** gray **=** 0**;**

uchar red **=** 0**,** green **=** 0**,** blue **=** 0**;**

**if(**bpp **==** 8**){**

uchar index **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**);**

uint location **=** 54 **+** index **\*** 4**;**

blue **=** getByte**(**fp**,** location**);**

green **=** getByte**(**fp**,** location **+** 1**);**

red **=** getByte**(**fp**,** location **+** 2**);**

**}** **else** **if(**bpp **==** 24**)** **{**

blue **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3**);**

green **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**\***3 **+** 1**);**

red **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**\***3 **+** 2**);**

**}**

gray **=** 0.3**\***red **+** 0.59**\***green **+** 0.11**\***blue**;**

**if(**gray **<=** 127**)**

**\*(**monochromeBits **+** i**\***newRowSizeInBits **+** j**)** **=** 0**;**

**else**

**\*(**monochromeBits **+** i**\***newRowSizeInBits **+** j**)** **=** 1**;**

**}**

**for(int** k **=** j**;** k **<** newRowSizeInBits**;** k**++)** **{**

**\*(**monochromeBits **+** i**\***newRowSizeInBits **+** j**)** **=** 0**;**

**}**

*// Print progress*

**int** progress **=** i**\***100.0**/**height**;**

**if(**progress **%** 5 **>=**4**)** **{**

system**("cls");**

printf**("Converting... Please wait... (Ctrl+C to break)\nProgress: %d %%",** progress**);**

**}**

**}**

**\***dataSize **=** newRowSizeInBits**\***height**/**8**;**

**char** **\***monochromeData **=** **(char** **\*)**calloc**(\***dataSize**,** **sizeof(char));**

**for(unsigned** **long** i **=** 0**;** i **<** **\***dataSize**;** i**++)** **{**

**int** byte**[**8**];**

**for(unsigned** **long** j **=** 0**;** j **<** 8**;** j**++)**

byte**[**j**]** **=** monochromeBits**[**8**\***i**+**j**];**

monochromeData**[**i**]** **=** BitToByte**(**byte**);**

**}**

**return** monochromeData**;**

**}**

### Make a bitmap binary image file array [4]

* The function create an entire bitmap file array including header, DIB header and pixel array based on the file format (VI).
* Four bytes values (eg. width, heigh, file size etc.) are created using ToFourByte function which described at section e.
* Input: bitmap array pointer, width, height, binary data pointer, data size
* Output: bitmap array

**void** BMPmake**(char** **\***bitmap**,** **int** width**,** **int** height**,** **char** **\***data**,** **int** dataSize**)** **{**

*// -- FILE HEADER -- //*

*// bitmap signature*

bitmap**[**0**]** **=** 'B'**;**

bitmap**[**1**]** **=** 'M'**;**

*// file size*

**unsigned** **long** fileSize **=** 62 **+** dataSize**;**

**char** fileSizeBytes**[**4**];**

ToFourBytes**(**fileSize**,** fileSizeBytes**);**

bitmap**[**2**]** **=** fileSizeBytes**[**3**];** *// 40 (dib) + 14 (header) + 12 (data)*

bitmap**[**3**]** **=** fileSizeBytes**[**2**];**

bitmap**[**4**]** **=** fileSizeBytes**[**1**];**

bitmap**[**5**]** **=** fileSizeBytes**[**0**];**

*// reserved field (in hex. 00 00 00 00)*

**for(int** i **=** 6**;** i **<** 10**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// offset of pixel data inside the image*

bitmap**[**10**]** **=** 62**;**

**for(int** i **=** 11**;** i **<** 14**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// -- DIB HEADER -- //*

*// header size*

bitmap**[**14**]** **=** 40**;**

**for(int** i **=** 15**;** i **<** 18**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// width of the image*

**char** widthBytes**[**4**];**

ToFourBytes**(**width**,** widthBytes**);**

bitmap**[**18**]** **=** widthBytes**[**3**];**

bitmap**[**19**]** **=** widthBytes**[**2**];**

bitmap**[**20**]** **=** widthBytes**[**1**];**

bitmap**[**21**]** **=** widthBytes**[**0**];**

*// height of the image*

**char** heightBytes**[**4**];**

ToFourBytes**(**height**,** heightBytes**);**

bitmap**[**22**]** **=** heightBytes**[**3**];**

bitmap**[**23**]** **=** heightBytes**[**2**];**

bitmap**[**24**]** **=** heightBytes**[**1**];**

bitmap**[**25**]** **=** heightBytes**[**0**];**

*// reserved*

bitmap**[**26**]** **=** 1**;**

bitmap**[**27**]** **=** 0**;**

*// number of bits per pixel*

bitmap**[**28**]** **=** 1**;** *// 1 bit*

bitmap**[**29**]** **=** 0**;**

*// compression method (no compression here)*

**for(int** i **=** 30**;** i **<** 34**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// size of pixel data*

**char** dataSizeBytes**[**4**];**

ToFourBytes**(**dataSize**,** dataSizeBytes**);**

bitmap**[**34**]** **=** dataSizeBytes**[**3**];**

bitmap**[**35**]** **=** dataSizeBytes**[**2**];**

bitmap**[**36**]** **=** dataSizeBytes**[**1**];**

bitmap**[**37**]** **=** dataSizeBytes**[**0**];**

*// unimportant*

**for(int** i **=** 38**;** i **<** 54**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// color table 0x00000000, 0xffffff00*

bitmap**[**54**]** **=** 0**;**

bitmap**[**55**]** **=** 0**;**

bitmap**[**56**]** **=** 0**;**

bitmap**[**57**]** **=** 0**;**

bitmap**[**58**]** **=** 0xff**;**

bitmap**[**59**]** **=** 0xff**;**

bitmap**[**60**]** **=** 0xff**;**

bitmap**[**61**]** **=** 0**;**

*// -- PIXEL DATA -- //*

**for(unsigned** **long** i **=** 62**;** i **<** fileSize**;** i**++)** **{**

bitmap**[**i**]** **=** data**[**i **-** 62**];**

**}**

**}**

### Write a new binary bitmap image [4]

* The function writes the bitmap file array (created from BMPmake) to a file.
* Input: bitmap array, file size, file location (file name included)

**void** BMPwrite**(char** **\***bitmap**,** **unsigned** **long** fileSize**,** **char** **\***location**)** **{**

FILE **\***file**;**

file **=** fopen**(**location**,** **"wb");**

**for(unsigned** **long** i **=** 0**;** i **<** fileSize**;** i**++)** **{**

fputc**(**bitmap**[**i**],** file**);**

**}**

fclose**(**file**);**

**}**

## UI related (Windows only)

* Used for printing the image.
* The functions will scale the window and font size for better printing.
  + Window size will be set maximum as the resolution (needed to press the maximize button).
  + Font size will be set as .
* Windows 7 partially supported (unstable)  
  Windows 8, Windows 8.1 untested  
  Windows 10, Windows 11 fully supported
* Libraries: stdio.h , windows.h, cwchar.

### char to wchar

* wChar is used for Unicode UTF-16 strings, a standard/native string encoding used in Win32.
* Input: char\*
* Output: wchar\_t\*

**static** **wchar\_t\*** charToWChar**(const** **char\*** text**)**

**{**

**const** size\_t size **=** strlen**(**text**)** **+** 1**;**

**wchar\_t\*** wText **=** **new** **wchar\_t[**size**];**

mbstowcs**(**wText**,** text**,** size**);**

**return** wText**;**

**}**

### Get screen resolution [5]

**void** GetDesktopResolution**(int&** horizontal**,** **int&** vertical**)** **{**

RECT desktop**;**

*// Get a handle to the desktop window*

**const** HWND hDesktop **=** GetDesktopWindow**();**

*// Get the size of screen to the variable desktop*

GetWindowRect**(**hDesktop**,** **&**desktop**);**

*// The top left corner will have coordinates (0,0)*

*// and the bottom right corner will have coordinates*

*// (horizontal, vertical)*

horizontal **=** desktop**.**right**;**

vertical **=** desktop**.**bottom**;**

**}**

### Get current font width

**int** GetCurrentFontWidth**(){**

CONSOLE\_FONT\_INFO fontInfo**;**

GetCurrentConsoleFont**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** FALSE**,** **&**fontInfo**);**

**return** fontInfo**.**dwFontSize**.**X**;**

**}**

### Set console window and font size [6] [7]

* Input: width, height, text width, text height, font name

**void** SetConsoleSize**(int** width**,** **int** height**,** **int** textWidth**,** **int** textHeight**,** **char\*** fontName**){**

*// Font size preset*

**int** defaultTextSize**[**3**][**2**]** **=** **{{**4**,** 6**},** **{**6**,** 8**},** **{**8**,** 8**}};**

COORD coord**;**

coord**.**X **=** width **+** 2**;** *// Defining our X and*

coord**.**Y **=** height **+** 2**;** *// Y size for buffer.*

SMALL\_RECT rect**;**

rect**.**Top **=** 0**;**

rect**.**Left **=** 0**;**

rect**.**Bottom **=** coord**.**Y**-**1**;** *// height for window*

rect**.**Right **=** coord**.**X**-**1**;** *// width for window*

HANDLE hwnd **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);** *// get handle*

SetConsoleScreenBufferSize**(**hwnd**,** coord**);** *// set buffer size*

SetConsoleWindowInfo**(**hwnd**,** TRUE**,** **&**rect**);** *// set window size*

CONSOLE\_FONT\_INFOEX cfi**;**

cfi**.**cbSize **=** **sizeof(**cfi**);**

cfi**.**nFont **=** 0**;**

cfi**.**dwFontSize**.**X **=** textWidth**;** *// Width of each character in the font*

cfi**.**dwFontSize**.**Y **=** textHeight**;** *// Height*

cfi**.**FontFamily **=** FF\_DONTCARE**;**

cfi**.**FontWeight **=** FW\_NORMAL**;**

std**::**wcscpy**(**cfi**.**FaceName**,** charToWChar**(**fontName**));**

SetCurrentConsoleFontEx**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** FALSE**,** **&**cfi**);**

*// If set fails (Windows 7), then set using font size preset*

**if(**GetCurrentFontWidth**()** **!=** textWidth**){**

**int** i **=** 0**;**

**while(**textWidth **>** defaultTextSize**[**i**][**0**]** **&&** i **<** 3**)** i**++;**

cfi**.**dwFontSize**.**X **=** defaultTextSize**[**i**][**0**];**

cfi**.**dwFontSize**.**Y **=** defaultTextSize**[**i**][**1**];**

SetCurrentConsoleFontEx**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** FALSE**,** **&**cfi**);**

**}**

**}**

### Set print size

* Font size will be set as .
* Input: image width, image height

**void** SetPrintSize**(int** width**,** **int** height**){**

**int** horizontal**,** vertical**;**

GetDesktopResolution**(**horizontal**,** vertical**);**

SetConsoleSize**(**width**,** height**,** horizontal**/**width**,** horizontal**/**width**,** **"Raster Fonts");**

**}**

### Set default console size

* Window width, height: 120, 30
* Font width, height: 8, 16
* Font name: Consolas

**void** SetDefaultConsoleSize**(){**

SetConsoleSize**(**120**,** 30**,** 8**,** 16**,** **"Consolas");**

**}**

# Complete code

## Main

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include "bmpLib.h"

#include "bmpWrite.h"

#include "UILib.h"

**char** **const** **\***title **=** **" \_\_\_\_\_\_\n\**

**\_\_\_\_\_\_\_\_\_ .---\"\"\" \"\"\"---.\n\**

**:\_\_\_\_\_\_.-': : .--------------. :\n\**

**| \_\_\_\_\_\_ | | : : |\n\**

**|:\_\_\_\_\_\_B:| | | Welcome to | |\n\**

**|:\_\_\_\_\_\_B:| | | | |\n\**

**|:\_\_\_\_\_\_B:| | | Image to | |\n\**

**| | | | Monochrome | |\n\**

**|:\_\_\_\_\_: | | | Converter | |\n\**

**| == | | : : |\n\**

**| O | : '--------------' :\n\**

**| o | :'---...\_\_\_\_\_\_...---'\n\**

**| o |-.\_.-i\_\_\_/' \\.\_\n\**

**|'-.\_\_\_\_o\_| '-. '-...\_\_\_\_\_\_...-' `-.\_\n\**

**:\_\_\_\_\_\_\_\_\_: `.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ `-.\_\_\_.-.\n\**

**.'.eeeeeeeeeeeeeeeeee.'. :\_\_\_:\n\**

**.'.eeeeeeeeeeeeeeeeeeeeee.'.\n\**

**:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:\n\n";**

**char** **const** **\***end **=** **" .-\"\"\"-.\n\**

**/ .===. \\\n\**

**\\/ 6 6 \\/\n\**

**( \\\_\_\_/ )\n\**

**\_\_\_\_\_\_ooo\_\_\\\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\n\**

**/Thanks for using my program! \\\n\**

**| Creator: Luong Trien Thang |\n\**

**| Semester - Class: 212 - TT03 |\n\**

**| Lecturer: Dang Thanh Tin |\n\**

**| Tran Anh Khoa |\n\**

**| 15 May 2022 |\n\**

**\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ooo\_\_\_\_\_\_\_\_/\n\**

**| | |\n\**

**|\_ | \_|\n\**

**| | |\n\**

**|\_\_|\_\_|\n\**

**/-'Y'-\\\n\**

**(\_\_/ \\\_\_)";**

**int** main**()** **{**

main**:** **{**

SetConsoleTitle**("Image to Monochrome");** *// Set console title*

system**("cls");** *// Clear screen*

SetDefaultConsoleSize**();** *// Set default console size*

fflush**(**stdin**);** *// Clear input buffer(s)*

printf**(**title**);** *// Print title art*

**char** pathInput**[**265**]** **=** **"";** *// path string*

**char** path**[**260**]** **=** **"";** *// real path*

printf**("Please input the file locaion (drag and drop is supported).\nType h or help to show the instructions.\nType q or quit to end the program.\n");**

gets**(**pathInput**);** *// get path string*

printf**("\n");**

**if(**strcmp**(**pathInput**,** **"h")** **==** 0 **||** strcmp**(**pathInput**,** **"help")** **==** 0**)** **{**

system**("cls");**

printf**(**title**);**

printf**("INSTRUCTIONS:\n");**

printf**("The program only accepts input of 8bpp and 24bpp bitmap image.\n");**

printf**("\n");**

printf**("For printing:\n");**

printf**("\tConsole and font will be resized for better viewing.\n");**

printf**("\tPress <Enter> once, then please maximize the window by clicking on the middle icon at the top right corner.\n");**

printf**("\tPress <Enter> again to print monochrome version of the image.\n");**

printf**("\tPress <Enter> after printing to restore console size and font.\n");**

printf**("\n");**

printf**("For converting:\n");**

printf**("\tThe image will be converted to monochrome bitmap file and saved at the same location as the input.\n");**

system**("pause");**

**goto** main**;**

**}** **else** **if(**strcmp**(**pathInput**,** **"q")** **==** 0 **||** strcmp**(**pathInput**,** **"quit")** **==** 0**)** **{**

**goto** end**;**

**}**

**if(**pathInput**[**0**]** **==** '"'**)** *// If the path contains " (double quotes), remove it then copy to path*

strncpy**(**path**,** pathInput **+** 1**,** strlen**(**pathInput**)** **-** 2**);**

**else**

strcpy**(**path**,** pathInput**);** *// Else, just copy*

FILE **\***fp**=**fopen**(**path**,"r");** *// Open file*

**if(**fp **==** NULL**)** **{** *// If file not found*

printf**("ERROR: File not found\n");**

**goto** restartPrompt**;**

**}**

**if(!**IsBitMap**(**fp**))** **{** *// Check whether a bitmap file*

printf**("ERROR: Format error. Or not a bitmap file!\n");**

fclose**(**fp**);**

**goto** restartPrompt**;**

**}**

**int** width **=** getWidth**(**fp**);** *// Get width, height, bpp*

**int** height **=** getHeight**(**fp**);**

**int** bpp **=** getBpp**(**fp**);**

printf**("File path: %s\nSize: %dpx x %dpx\nBit per pixel: %d\n",** path**,** width**,** height**,** bpp**);**

**if(**bpp **==** 1**){** *// If bpp = 1 (monochrome, binary image)*

printf**("ERROR: The file is already in monochrome (1bpp).\n");**

fclose**(**fp**);**

**goto** restartPrompt**;**

**}**

**else** **if(**bpp **!=** 8 **&&** bpp **!=** 24**)** **{** *// Else if bpp is not 8 or 24*

printf**("ERROR: The file is not in 8bpp or 24bpp.\n");**

fclose**(**fp**);**

**goto** restartPrompt**;**

**}**

printf**("Do you want to print image or convert to monochrome? (p: print, c: convert) ");**

ask**:**

**char** pc **=** getch**();** *// Get character from keyboard*

printf**("\n");**

**if(**pc **==** 'p'**)** **{** *// PRINT!*

system**("pause");**

SetPrintSize**(**width**,** height**);** *// Set window and font size*

getch**();**

printImage**(**fp**,** width**,** height**,** bpp**);** *// Print binary image to console*

getch**();**

system**("cls");** *// Print end, clear screen*

SetDefaultConsoleSize**();** *// Revert console size when done*

**}** **else** **if** **(**pc **==** 'c'**)** **{** *// CONVERT!*

**unsigned** **long** dataSize**;**

**char** **\***dataPtr**;**

dataPtr **=** GetMonochromeData**(**fp**,** width**,** height**,** bpp**,** **&**dataSize**);** *// Get monochrome data*

**unsigned** **long** fileSize **=** 62 **+** dataSize**;** *// Filesize = 62 (file header + DIB header) + dataSize*

**char** bitmap**[**fileSize**];**

**char** outputLocation**[**260**]** **=** **"";**

strncpy**(**outputLocation**,** path**,** strlen**(**path**)** **-** 4**);** *// Remove ".bmp" at the end of the path*

strcat**(**outputLocation**,** **"\_mono.bmp");** *// Add "\_mono.bmp" to the end of the path*

BMPmake**(**bitmap**,** width**,** height**,** dataPtr**,** dataSize**);** *// Make a bitmap file array*

BMPwrite**(**bitmap**,** fileSize**,** outputLocation**);** *// Write the array to the location*

free**(**dataPtr**);** *// Free the bitmap array*

system**("cls");**

printf**("Conversion completed. Your file is located at\n%s",** outputLocation**);**

printf**("\n");**

**}** **else** **{** *// If not "p" or "c"*

printf**("Please type p to print or c to convert. ");**

**goto** ask**;**

**}**

**}**

restartPrompt**:{**

printf**("Restart program? (y/[n]) ");**

**if(**getch**()** **==** 'y'**)**

**goto** main**;**

**}**

end**:** **{**

system**("cls");**

printf**(**end**);**

**}**

**return** 0**;**

**}**

## bmpLib

For read and process image.

### bmpLib.h

#include "bmpLib.cpp"

**int** IsBitMap**(**FILE **\***fp**);**

**int** getWidth**(**FILE **\***fp**);**

**int** getHeight**(**FILE **\***fp**);**

ushort getBit**(**FILE **\***fp**);**

uint getOffSet**(**FILE **\***fp**);**

uchar getByte**(**FILE **\***fp**,** **int** location**);**

**int** printImage**(**FILE **\***fp**,** **int** width**,** **int** height**);**

### bmpLib.cpp

#include <stdio.h>

#include <malloc.h>

#include <iostream>

#define BM 19778

#define ushort unsigned short

#define uint unsigned int

#define uchar unsigned char

*//Judge whether it is a bitmap, in 0-1 bytes*

**int** IsBitMap**(**FILE **\***fp**)** **{**

ushort s**;**

fread**(&**s**,**1**,**2**,**fp**);**

**return** s**==**BM **?** 1 **:** 0**;**

**}**

*//Get the width of the picture, in 18-21 bytes*

**int** getWidth**(**FILE **\***fp**)** **{**

**int** width**;**

fseek**(**fp**,**18**,**SEEK\_SET**);**

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

**return** width**;**

**}**

*//Get the height of the picture, in 22-25 bytes*

**int** getHeight**(**FILE **\***fp**)** **{**

**int** height**;**

fseek**(**fp**,**22**,**SEEK\_SET**);**

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

**return** height**;**

**}**

*//Get the number of bits of each pixel in 28-29 bytes*

ushort getBit**(**FILE **\***fp**)** **{**

ushort bit**;**

fseek**(**fp**,**28**,**SEEK\_SET**);**

fread**(&**bit**,**1**,**2**,**fp**);**

**return** bit**;**

**}**

*//Get the starting position of data, in 10-13 bytes*

uint getOffSet**(**FILE **\***fp**)** **{**

uint OffSet**;**

fseek**(**fp**,**10L**,**SEEK\_SET**);**

fread**(&**OffSet**,**1**,**4**,**fp**);**

**return** OffSet**;**

**}**

*//Get the byte in a location*

uchar getByte**(**FILE **\***fp**,** **int** location**)** **{**

uchar byte**;**

fseek**(**fp**,**location**,**SEEK\_SET**);**

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

**return** byte**;**

**}**

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

**int** i**,**j**;**

**int** size **=** width **\*** height**;**

uint offset **=** getOffSet**(**fp**);**

ushort bit **=** getBit**(**fp**);**

**int** rowSize **=** **int((**bpp**\***width**\***1.0 **+** 31**)/**32**)\***4**;**

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

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

**float** gray **=** 0**;**

uchar red **=** 0**,** green **=** 0**,** blue **=** 0**;**

**if(**bpp **==** 8**){**

uchar index **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**);**

uint location **=** 54 **+** index **\*** 4**;**

blue **=** getByte**(**fp**,** location**);**

green **=** getByte**(**fp**,** location **+** 1**);**

red **=** getByte**(**fp**,** location **+** 2**);**

**}** **else** **if(**bpp **==** 24**)** **{**

blue **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3**);**

green **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3 **+** 1**);**

red **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3 **+** 2**);**

**}**

gray **=** 0.3**\***red **+** 0.59**\***green **+** 0.11**\***blue**;**

**if(**gray **<=** 127**)**

printf**("0");**

**else**

printf**("1");**

**}**

printf**("\n");**

**}**

**}**

**char** BitToByte**(int** **\***num**)** **{**

**char** result **=** 0**;**

**for** **(int** i **=** 0**;** i **<** 8**;** **++**i **)**

result **|=** **(**num**[**i**]** **==** 1**)** **<<** **(**7 **-** i**);**

**return** result**;**

**}**

**char** **\***GetMonochromeData**(**FILE **\***fp**,** **int** width**,** **int** height**,** **int** bpp**,** **unsigned** **long** **\***dataSize**)** **{**

uint offset **=** getOffSet**(**fp**);**

**int** rowSize **=** **int((**bpp**\***width**\***1.0 **+** 31**)/**32**)\***4**;**

**unsigned** **long** newRowSizeInBits **=** **int((**1**\***width**\***1.0 **+** 31**)/**32**)\***4**\***8**;**

**char** **\***monochromeBits **=** **(char** **\*)**calloc**(**newRowSizeInBits **\*** height**,** **sizeof(char));**

**int** progressCounter **=** 5**;**

system**("cls");**

printf**("Converting... Please wait... (Ctrl+C to break)\nProgress: 0 %%");**

**for(int** i **=** 0**;** i **<** height**;** i**++)** **{**

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

**float** gray **=** 0**;**

uchar red **=** 0**,** green **=** 0**,** blue **=** 0**;**

**if(**bpp **==** 8**){**

uchar index **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**);**

uint location **=** 54 **+** index **\*** 4**;**

blue **=** getByte**(**fp**,** location**);**

green **=** getByte**(**fp**,** location **+** 1**);**

red **=** getByte**(**fp**,** location **+** 2**);**

**}** **else** **if(**bpp **==** 24**)** **{**

blue **=** getByte**(**fp**,** offset **+** i**\***rowSize **+** j**\***3**);**

green **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**\***3 **+** 1**);**

red **=** getByte**(**fp**,** offset**+** i**\***rowSize **+** j**\***3 **+** 2**);**

**}**

gray **=** 0.3**\***red **+** 0.59**\***green **+** 0.11**\***blue**;**

**if(**gray **<=** 127**)**

**\*(**monochromeBits **+** i**\***newRowSizeInBits **+** j**)** **=** 0**;**

**else**

**\*(**monochromeBits **+** i**\***newRowSizeInBits **+** j**)** **=** 1**;**

**}**

**for(int** k **=** j**;** k **<** newRowSizeInBits**;** k**++)** **{**

**\*(**monochromeBits **+** i**\***newRowSizeInBits **+** j**)** **=** 0**;**

**}**

*// Print progress*

**int** progress **=** i**\***100.0**/**height**;**

**if(**progress **%** 5 **>=**4**)** **{**

system**("cls");**

printf**("Converting... Please wait... (Ctrl+C to break)\nProgress: %d %%",** progress**);**

**}**

**}**

**\***dataSize **=** newRowSizeInBits**\***height**/**8**;**

**char** **\***monochromeData **=** **(char** **\*)**calloc**(\***dataSize**,** **sizeof(char));**

**for(unsigned** **long** i **=** 0**;** i **<** **\***dataSize**;** i**++)** **{**

**int** byte**[**8**];**

**for(unsigned** **long** j **=** 0**;** j **<** 8**;** j**++)**

byte**[**j**]** **=** monochromeBits**[**8**\***i**+**j**];**

monochromeData**[**i**]** **=** BitToByte**(**byte**);**

**}**

**return** monochromeData**;**

**}**

## bmpWrite

For writing bitmap file.

### bmpWrite.h

#include "bmpWrite.cpp"

**void** BMPmake**(char** **\***bitmap**,** **int** width**,** **int** height**,** **char** **\***data**,** **int** dataSize**);**

**void** BMPwrite**(char** **\***bitmap**,** **int** fileSize**);**

**void** WriteFile**(char** **\***data**);**

### bmpWrite.cpp

#include <stdio.h>

**void** ToFourBytes**(unsigned** **long** n**,** **char** bytes**[**4**]){**

bytes**[**0**]** **=** **(**n **>>** 24**)** **&** 0xFF**;**

bytes**[**1**]** **=** **(**n **>>** 16**)** **&** 0xFF**;**

bytes**[**2**]** **=** **(**n **>>** 8**)** **&** 0xFF**;**

bytes**[**3**]** **=** **(**n **>>** 0**)** **&** 0xFF**;**

**}**

**void** BMPmake**(char** **\***bitmap**,** **int** width**,** **int** height**,** **char** **\***data**,** **int** dataSize**)** **{**

*// -- FILE HEADER -- //*

*// bitmap signature*

bitmap**[**0**]** **=** 'B'**;**

bitmap**[**1**]** **=** 'M'**;**

*// file size*

**unsigned** **long** fileSize **=** 62 **+** dataSize**;**

**char** fileSizeBytes**[**4**];**

ToFourBytes**(**fileSize**,** fileSizeBytes**);**

bitmap**[**2**]** **=** fileSizeBytes**[**3**];** *// 40 (dib) + 14 (header) + 12 (data)*

bitmap**[**3**]** **=** fileSizeBytes**[**2**];**

bitmap**[**4**]** **=** fileSizeBytes**[**1**];**

bitmap**[**5**]** **=** fileSizeBytes**[**0**];**

*// reserved field (in hex. 00 00 00 00)*

**for(int** i **=** 6**;** i **<** 10**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// offset of pixel data inside the image*

bitmap**[**10**]** **=** 62**;**

**for(int** i **=** 11**;** i **<** 14**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// -- DIB HEADER -- //*

*// header size*

bitmap**[**14**]** **=** 40**;**

**for(int** i **=** 15**;** i **<** 18**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// width of the image*

**char** widthBytes**[**4**];**

ToFourBytes**(**width**,** widthBytes**);**

bitmap**[**18**]** **=** widthBytes**[**3**];**

bitmap**[**19**]** **=** widthBytes**[**2**];**

bitmap**[**20**]** **=** widthBytes**[**1**];**

bitmap**[**21**]** **=** widthBytes**[**0**];**

*// height of the image*

**char** heightBytes**[**4**];**

ToFourBytes**(**height**,** heightBytes**);**

bitmap**[**22**]** **=** heightBytes**[**3**];**

bitmap**[**23**]** **=** heightBytes**[**2**];**

bitmap**[**24**]** **=** heightBytes**[**1**];**

bitmap**[**25**]** **=** heightBytes**[**0**];**

*// reserved*

bitmap**[**26**]** **=** 1**;**

bitmap**[**27**]** **=** 0**;**

*// number of bits per pixel*

bitmap**[**28**]** **=** 1**;** *// 1 bit*

bitmap**[**29**]** **=** 0**;**

*// compression method (no compression here)*

**for(int** i **=** 30**;** i **<** 34**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// size of pixel data*

**char** dataSizeBytes**[**4**];**

ToFourBytes**(**dataSize**,** dataSizeBytes**);**

bitmap**[**34**]** **=** dataSizeBytes**[**3**];**

bitmap**[**35**]** **=** dataSizeBytes**[**2**];**

bitmap**[**36**]** **=** dataSizeBytes**[**1**];**

bitmap**[**37**]** **=** dataSizeBytes**[**0**];**

*// unimportant*

**for(int** i **=** 38**;** i **<** 54**;** i**++)** bitmap**[**i**]** **=** 0**;**

*// color table 0x00000000, 0xffffff00*

bitmap**[**54**]** **=** 0**;**

bitmap**[**55**]** **=** 0**;**

bitmap**[**56**]** **=** 0**;**

bitmap**[**57**]** **=** 0**;**

bitmap**[**58**]** **=** 0xff**;**

bitmap**[**59**]** **=** 0xff**;**

bitmap**[**60**]** **=** 0xff**;**

bitmap**[**61**]** **=** 0**;**

*// -- PIXEL DATA -- //*

**for(unsigned** **long** i **=** 62**;** i **<** fileSize**;** i**++)** **{**

bitmap**[**i**]** **=** data**[**i **-** 62**];**

**}**

**}**

**void** BMPwrite**(char** **\***bitmap**,** **unsigned** **long** fileSize**,** **char** **\***location**)** **{**

FILE **\***file**;**

file **=** fopen**(**location**,** **"wb");**

**for(unsigned** **long** i **=** 0**;** i **<** fileSize**;** i**++)** **{**

fputc**(**bitmap**[**i**],** file**);**

**}**

fclose**(**file**);**

**}**

## UILib

### UILib.h

#include "UILib.cpp"

**void** GetDesktopResolution**(int&** horizontal**,** **int&** vertical**);**

**void** SetConsoleSize**();**

**void** SetDefaultConsoleSize**();**

**void** SetPrintSize**(int** width**,** **int** height**);**

**int** GetCurrentFontWidth**();**

### UILib.cpp

#include <windows.h>

#include <stdio.h>

#include <cwchar>

#include "wtypes.h"

**void** GetDesktopResolution**(int&** horizontal**,** **int&** vertical**)** **{**

RECT desktop**;**

*// Get a handle to the desktop window*

**const** HWND hDesktop **=** GetDesktopWindow**();**

*// Get the size of screen to the variable desktop*

GetWindowRect**(**hDesktop**,** **&**desktop**);**

*// The top left corner will have coordinates (0,0)*

*// and the bottom right corner will have coordinates*

*// (horizontal, vertical)*

horizontal **=** desktop**.**right**;**

vertical **=** desktop**.**bottom**;**

**}**

**static** **wchar\_t\*** charToWChar**(const** **char\*** text**)**

**{**

**const** size\_t size **=** strlen**(**text**)** **+** 1**;**

**wchar\_t\*** wText **=** **new** **wchar\_t[**size**];**

mbstowcs**(**wText**,** text**,** size**);**

**return** wText**;**

**}**

**int** GetCurrentFontWidth**(){**

CONSOLE\_FONT\_INFO fontInfo**;**

GetCurrentConsoleFont**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** FALSE**,** **&**fontInfo**);**

**return** fontInfo**.**dwFontSize**.**X**;**

**}**

**void** SetConsoleSize**(int** width**,** **int** height**,** **int** textWidth**,** **int** textHeight**,** **char\*** fontName**){**

**int** defaultTextSize**[**3**][**2**]** **=** **{{**4**,** 6**},** **{**6**,** 8**},** **{**8**,** 8**}};**

COORD coord**;**

coord**.**X **=** width **+** 2**;** *// Defining our X and*

coord**.**Y **=** height **+** 2**;** *// Y size for buffer.*

SMALL\_RECT rect**;**

rect**.**Top **=** 0**;**

rect**.**Left **=** 0**;**

rect**.**Bottom **=** coord**.**Y**-**1**;** *// height for window*

rect**.**Right **=** coord**.**X**-**1**;** *// width for window*

HANDLE hwnd **=** GetStdHandle**(**STD\_OUTPUT\_HANDLE**);** *// get handle*

SetConsoleScreenBufferSize**(**hwnd**,** coord**);** *// set buffer size*

SetConsoleWindowInfo**(**hwnd**,** TRUE**,** **&**rect**);** *// set window size*

CONSOLE\_FONT\_INFOEX cfi**;**

cfi**.**cbSize **=** **sizeof(**cfi**);**

cfi**.**nFont **=** 0**;**

cfi**.**dwFontSize**.**X **=** textWidth**;** *// Width of each character in the font*

cfi**.**dwFontSize**.**Y **=** textHeight**;** *// Height*

cfi**.**FontFamily **=** FF\_DONTCARE**;**

cfi**.**FontWeight **=** FW\_NORMAL**;**

std**::**wcscpy**(**cfi**.**FaceName**,** charToWChar**(**fontName**));**

SetCurrentConsoleFontEx**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** FALSE**,** **&**cfi**);**

**if(**GetCurrentFontWidth**()** **!=** textWidth**){**

**int** i **=** 0**;**

**while(**textWidth **>** defaultTextSize**[**i**][**0**]** **&&** i **<** 3**)** i**++;**

cfi**.**dwFontSize**.**X **=** defaultTextSize**[**i**][**0**];**

cfi**.**dwFontSize**.**Y **=** defaultTextSize**[**i**][**1**];**

SetCurrentConsoleFontEx**(**GetStdHandle**(**STD\_OUTPUT\_HANDLE**),** FALSE**,** **&**cfi**);**

**}**

**}**

**void** SetPrintSize**(int** width**,** **int** height**){**

**int** horizontal**,** vertical**;**

GetDesktopResolution**(**horizontal**,** vertical**);**

SetConsoleSize**(**width**,** height**,** horizontal**/**width**,** horizontal**/**width**,** **"Raster Fonts");**

**}**

**void** SetDefaultConsoleSize**(){**

SetConsoleSize**(**120**,** 30**,** 8**,** 16**,** **"Consolas");**

**}**

# Appendix

The source code will be available on Github after 15th May, 2022 via this [link](https://github.com/superzeldalink/bmp2monochrome/tree/code).

# References

|  |  |
| --- | --- |
| [1] | Wikipedia, "BMP file format - Wikipedia," [Online]. Available: <https://en.wikipedia.org/wiki/BMP_file_format>. |
| [2] | tutorialspoint, "Grayscale to RGB Conversion," [Online]. Available: <https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm>. |
| [3] | marcnyc, "Reading bmp image with C language," 24 March 2022. [Online]. Available: <https://programs.wiki/wiki/reading-bmp-image-with-c-language.html>. |
| [4] | Devos50, "Creating a BMP file (bitmap) in C - Stack Overflow," 12 June 2012. [Online]. Available: <https://stackoverflow.com/questions/11004868/creating-a-bmp-file-bitmap-in-c>. |
| [5] | cppkid, "How To Get The Screen Resolution In Pixels," [Online]. Available: <https://cppkid.wordpress.com/2009/01/07/how-to-get-the-screen-resolution-in-pixels/>. |
| [6] | Krever, "c++ - Reducing console size," 15 October 2012. [Online]. Available: [https://stackoverflow.com/questions/12900713/reducing-console-size/12901833#12901833](https://stackoverflow.com/questions/12900713/reducing-console-size/12901833%2312901833). |
| [7] | A. DM, "c++ - How to change the console font size," 13 Feburary 2016. [Online]. Available: <https://stackoverflow.com/questions/35382432/how-to-change-the-console-font-size>. |