10th MODULE PRACTICUM REPORT

BINARY FILE OPERATION

ALGORITHM AND PROGRAMMING LAB

Lecturer :

I Ketut Purnamawan, S.Kom., M.Kom.

Arranged by :

Ni Putu Karisma Dewi (2115101059)

COMPUTER SCIENCE STUDY PROGRAM
DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF ENGINEERING AND VOCATIONAL
UNIVERSITAS PENDIDIKAN GANESHA
2021/2022

A. Problem

Create a program to process images with Windows BMP RGB24 file format. The program can create an image that is lighter than the original image.

B. Theory

- C Language

The C programming language was first created by Dennis Ritchie in 1972 to program computer systems and networks. Although initially functioned to do system programming, C language can also be used in developing application software.

The C++ language itself is an extension of the C language itself. C++ was created by Bjarne Stroustrup which was developed at Bell Labs (Dennis Ritchie) in the early 1970s. Bjarne Stroustrup at Bell Labs first developed C++ in the early 1980s. To support the features in C++, built efficiency and support system for low-level programming (low level coding). In C++ added new concepts such as classes with properties such as inheritance and overloading. One of the most fundamental differences with the C language is its support for object-oriented programming concepts.

- Header

  The header file has a function to call and execute the functions contained in the C++ header file library for use in a C++ file. In this program, two header files are used, namely:

  - ✓ #include<cstido> : function to call macro functions and input output operations

- Data Type

  The data type represents the type of value that is contained in the program. When declaring a variable, the data type of the variable must be specified. Errors in providing the data type will result in the program not being performed. Even if the software can be executed, it will produce incorrect results, as predicted. Data types in C are classified into two sorts, namely basic data types and constructed data types. Examples of basic data types are integer, char, bool, float, etc. While examples of formed data types are structures, arrays, strings, etc.

- Branching Algorithm

- If : If the condition is true, then the command will be executed and if it does not meet the conditions it will be ignored. (syntax: if(condition))
- If else : If the condition evaluates to true, then command-1 will be executed and if it doesn't meet the conditions it will execute command-2, and the next command if the previous command doesn't meet the conditions.

- Switch case

Switch case statement evaluates a given expression and based on the evaluated value (matching a certain condition), it executes the statements associated with it. Basically, it is used to perform different actions based on different conditions(cases).

- Switch case statements follow a selection-control mechanism and allow a value to change control of execution.
- They are a substitute for long if statements that compare a variable to several integral values.
- The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

```
switch(edit)
                        {
                            case 1:
                            {
                                cout<<"Insert name : ";
                                getline(cin, array[ID].name);
                                getline(cin, array[ID].name);
                                break;
                            }
                            case 2:
                            {
                                cout<<"Insert Gender : ";
                                cin>>array[ID].gender;
                                break;

                            }
                            case 3:
                            {
                                cout<<"Insert Year of birth : ";
                                cin>>array[ID].yearOfBirth;
                                break;
                            }
                            case 4:
                            {
```

```
        cout<<"Insert of addmission : ";
        cin>>array[ID].yearIn;
        break;
    }
    case 5 :
    {
        cout<<"Insert Student's GPA : ";
        cin>>array[ID].gpa;
        break;
    }
    case 6 :
    {
        cout<<"Insert all student data";
        array[ID]=input();

        break;
    }
    default :
    {
        cout<<"Invalid input";
        break;
    }
```

Some important keywords:

- Break: This keyword is used to stop the execution inside a switch block. It helps to terminate the switch block and break out of it.
- Default: This keyword is used to specify the set of statements to execute if there is no case match.

- Function

  A function is a set of statements that take inputs, do some specific computation and produces output.

  The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

  A function declaration tells the compiler about the number of parameters function takes, data-types of parameters, and return type of function. Putting parameter names in function declaration is optional in the function declaration, but it is necessary to put them in the definition.

```
int findID(studentData array[], int length, string ID)
```

```
{

    for(int i=0; i<length; i++)
    {
        if(array[i].studentID==ID)
        return i;
    }
    return length+1; //if the function return higher than length, then
the ID is not found
}
```

- Array

  An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).

- Struct

  Array of struct A structure is a key word that create user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

- Array of struct

  An array is a collection of data items of the same type. Each element of the array can be int, char, float, double, or even a structure. We have seen that a structure allows elements of different data types to be grouped together under a single name. This structure can then be thought of as a new data type in itself. So, an array can comprise elements of this new data type. An array of structures finds its applications in grouping the records together and provides for fast accessing.

- File operation (file handling)

  File operation is a process to read and write from and to a file, either in text or binary. The operations that you can perform on a File in C are creating a new file, opening an existing file, reading data from an existing file, writing data to file, moving data a file, moving data to a specific location on the file, and closing the file.

  There are two different type of file handling in C, text file and binary file. Text file is file created with a .txt extension and using any simple text editor, for example notepad or excel. A text file stores information in the form of ASCII characters internally, but when the file opened, there are context of the text readable to humans. And binary file

is created with .bin extension. Binary file stores the information in the same way as the information is held in computer memory. Binary file much easier to access than text file. The C programming offers various operations associated with file handling, they are :

- Creating a new file or opening an existing file in your system : fopen()
- Closing a file : fclose()
- Reading characters from a line : getc()
- Writing characters in a file : putc()
- Reading a set of data from a file : fscanf()
- Writing a set of data in a file : fprintf()
- Reading and integral value from a file : getw()
- Writing an integral value in a file : putw()
- Setting a desired position in the file : fseek()
- Getting the current position in the file : ftell()
- Setting the position at the beginning point : rewind()

The basic syntax of opening a file : *fpointer = FILE *fopen(cost char *file_name, const char *mode);

Description :

- *fpointer : is the pointer to the file that establishes a connection between the file and the program
- *file_name : is the name of the file.
- *mode : is the mode in which we want to open our file.

The following modes in which the file can be opened are:

MODE	ELUCIDATION
r	We use it to open a text file in reading mode
w	We use it to open or create a text file in writing mode
a	We use it to open a text file in append mode
r+	We use it to open a text file in both reading and writing mode
w+	We use it to open a text file in both reading and writing mode
a+	We use it to open a text file in both reading and writing mode
rb	We use it to open a binary file in reading mode
wb	We use it to open or create a binary file in writing mode

ab	We use it to open a binary file in append mode
rb+	We use it to open a binary file in both reading and writing mode
wb+	We use it to open a binary file in both reading and writing mode
ab+	We use it to open a binary file in both reading and writing mode

- RGB color space

  An RGB color space is any additive color space based on the RGB color model. An RGB color space is defined by chromaticity coordinates of the red, green, and blue additive primaries, the white point which is usually a standard illuminant, and the transfer function which is also known as the tone response curve (TRC) or gamma.

RGB color spaces

Color space	Standard	Year	White point	Primaries						Display gamma	Transfer function parameters				
				Red		Green		Blue			γ	α	β	δ	βδ
				x_R	y_R	x_G	y_G	x_B	y_B	EOTF		a + 1	$K_0/\varphi = E_t$	φ	K_0
ISO RGB			floating	floating											
Extended ISO RGB															
sRGB	IEC 61966-2-1	1996, 1999	D65	0.64	0.33	0.30	0.60	0.15	0.06	2.2	$\frac{12}{5}$	1.055	0.0031308	12.92	0.04045
HDTV	ITU-R BT.709	1999								2.4	$\frac{20}{9}$	1.099	0.004	4.5	0.018
scRGB	IEC 61966-2-2	2003													
Adobe RGB		1998				0.21	0.71			2.2	$\frac{563}{256}$				
PAL / SECAM	EBU 3213-E, ITU-R BT.470/601 (B/G)	1970				0.29	0.60			2.8	$\frac{14}{5}$				
Apple RGB				0.625	0.34	0.28	0.595	0.155	0.07	1.8					
NTSC, MUSE	SMPTE RP 145 (C), 170M, 240M	1987		0.63		0.31				2.5	$\frac{20}{9}$	1.1115	0.0057	4	0.0228
NTSC-J			D93												
NTSC-FCC	ITU-R BT.470/601 (M)	1953	C	0.67	0.33	0.21	0.71	0.14	0.08	2.5	$\frac{11}{5}$				
PAL-M	ITU-R BT.470-6[4]	1972								2.2					
eciRGB	ISO 22028-4	2008, 2012	D50							1.8	3	1.16	0.008856	9.033	0.08
DCI-P3	SMPTE RP 431-2	2011	6300K	0.68	0.32	0.265	0.69	0.15	0.06	2.6	$\frac{13}{5}$				
Display P3	SMPTE EG 432-1	2010	D65							~2.2	$\frac{12}{5}$	1.055	0.0031308	12.92	0.04045
UHDTV	ITU-R BT.2020, BT.2100	2012, 2016		0.708	0.292	0.170	0.797	0.131	0.046	2.4		1.0993	0.018054	4.5	0.081243
Wide Gamut	(Adobe)		D50	0.735	0.265	0.115	0.826	0.157	0.018	2.2	$\frac{563}{256}$				
RIMM	ISO 22028-3	2006, 2012		0.7347	0.2653	0.1596	0.8404	0.0366	0.0001	2.222	$\frac{20}{9}$	1.099	0.0018	5.5	0.099
ProPhoto (ROMM)	ISO 22028-2	2006, 2013								1.8	$\frac{9}{5}$	1	0.001953125	16	0.031248
CIE RGB	CIE 1931 color space	1931	E			0.2738	0.7174	0.1666	0.0089						
CIE XYZ				1	0	0	1	0	0		1				
MAC	ITU-R BO.650-2[5]	1985	D65	0.67	0.33	0.21	0.71	0.14	0.08	2.8					

- BMP 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.

  The BMP file format is capable of storing two-dimensional digital images both monochrome and color, in various color depths, and optionally with data compression, alpha channels, and color profiles. The Windows Metafile(WMF) specification covers the BMP file format.

Windows Bitmap

Filename extension	`.bmp` , `.dib`
Internet media type	`image/bmp` [1] `image/x-bmp`

Type code	`'BMP '`
	`'BMPf'`
	`'BMPp'`

Uniform Type Identifier (UTI)	com.microsoft.bmp
Developed by	Microsoft Corporation
Type of format	Raster graphics
Open format?	OSP for WMF

Microsoft has defined a particular representation of color bitmaps of different color depths, as an aid to exchanging bitmaps between devices and applications with a variety of internal representations. They called these device-independent bitmaps or DIBs, and the file format for them is called DIB file format or BMP image file format.

According to Microsoft support. A device-independent bitmap (DIB) is a format used to define device-independent bitmaps in various color resolutions. The main purpose of DIBs is to allow bitmaps to be moved from one device to another (hence, the device-independent part of the name). A DIB is an external format, in contrast to a device-dependent bitmap, which appears in the system as a bitmap object (created by an application...). A DIB is normally transported in metafiles (usually using the StretchDIBits() function), BMP files, and the Clipboard (CF_DIB data format).

The following sections discuss the data stored in the BMP file or DIB in detail. This is the standard BMP file format. Some applications create bitmap image files which are not compliant with the Microsoft documentation. Also, not all fields are used; a value of 0 will be found in these unused fields.

Structure name	Optional	Size	Purpose	Comments
Bitmap file header	No	14 bytes	To store general information about the bitmap image file	Not needed after the file is loaded in memory
DIB header	No	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
Extra bit masks	Yes	3 or 4 DWORDs[6] (12 or 16 bytes)	To define the pixel format	Present only in case the DIB header is the `BITMAPINFOHEADER` and the Compression Method member is set to either BI_BITFIELDS or BI_ALPHABITFIELDS
Color table	Semi-optional	Variable size	To define colors used by the bitmap image data (Pixel array)	Mandatory for color depths ≤ 8 bits
Gap1	Yes	Variable size	Structure alignment	An artifact of the File offset to Pixel array in the Bitmap file header
Pixel array	No	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
Gap2	Yes	Variable size	Structure alignment	An artifact of the ICC profile data offset field in the DIB header
ICC color profile	Yes	Variable size	To define the color profile for color management	Can also contain a path to an external file containing the color profile. When loaded in memory as "non-packed DIB", it is located between the color table and Gap1.[7]

C. Solution, testing, and result

- There are three include that used in this program.

```
#include <iostream>
#include <string>
using namespace std;
```

- I created type data f1 and f1out with pointer ‘file’. Data type integer for character c, and i. pointer char *words to saving a sentence, and word for saving a character.

```
FILE* fl;
FILE* flOut;
unsigned char varByte;
```

- Create a input and output with string data type so the user can input the BMP path that wants to be edited. The BMP path that wants to input must the same place where the program is saved.

```
string input;
cout<<"Input the file name : ";
cin>>input;

cout<<"The format file must bmp and located in the same disk with
this program file\n"<<endl;

const char *file = &input[0];

//name when outputting the file
string output;
cout<<"The output file name : ";
cin>>output;

const char *fileOutput= &output[0];

fl = fopen(file, "rb");
flOut = fopen(fileOutput, "wb");
```

- After the file have been inputted, create a data type that use to how much the RGB value is increased. The range is 1 until 255 (the max value)

```
int lighter;
cout<<"increasing brightness [1-255] :  ";
cin>>lighter;
```

- Create for loop 54 times to copy the entire header for the input file so the output file which is 54 byte in size.

```
for (int i=0; i<54; i++)
{
    fread(&varByte, 1, 1, fl);
    fwrite(&varByte, 1, 1, flOut);
}
```

- Fread is a function to read from a file which consists of 4 arguments the first argument is the memory address to which the value will be store, the second argument the size of the element to read, the third argument the number of element to read, and the last one is the file pointer that the program read from. After that, there are two conditions which is is the when to value of the varByte that already include the lighter is greater that 255, so the output will stop at 255. And if that conditions is not satisfy, make output where the new value of varbyte is equal to varbyte plus lighter. And at the last, close the f1 and f1Output file so that disappear from the memory.

```
while (fread(&varByte, 1, 1, fl))
{
    if((varByte+lighter)>255)
    {
        varByte = 255;

    }
    else
    {
        varByte = varByte+lighter;
    }
    fwrite(&varByte, 1, 1, flOut);
}

fclose(fl);
fclose(flOut);
return 0;
```

D. Source code

```
#include <iostream>
#include <string>
using namespace std;

FILE* fl;
```

```
FILE* flOut;
unsigned char varByte;

int main()
{
    //name when input the file

    cout<<"\n\n10th Program : Binary File Operation\n";
    cout<<"Name : Ni Putu Karisma Dewi\n";
    cout<<"Student ID : 2115101059\n";


    cout<<"\nThe format file must bmp and located in the same disk with
this program file\n\n"<<endl;
    string input;
    cout<<"Input the file name : ";
    cin>>input;



    const char *file = &input[0];

    //name when outputting the file
    string output;
    cout<<"\nThe output file name : ";
    cin>>output;


    const char *fileOutput= &output[0];

    fl = fopen(file, "rb");
    flOut = fopen(fileOutput, "wb");

    int lighter;

    cout<<"\nincreasing brightness [1-255] :  ";
    cin>>lighter;


    for (int i=0; i<54; i++)
    {
        fread(&varByte, 1, 1, fl);
        fwrite(&varByte, 1, 1, flOut);
    }

    while (fread(&varByte, 1, 1, fl))
    {
        if((varByte+lighter)>255)
```

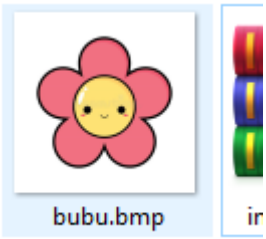
```
            {
                varByte = 255;

            }
            else
            {
                varByte = varByte+lighter;
            }
            fwrite(&varByte, 1, 1, flOut);
        }

        fclose(fl);
        fclose(flOut);
        return 0;
    }
```

Result

- We will use this file and the program is saved at local disk data.




```
PS C:\Users\LENOVO> cd "d:\KULIAH SMT 2\latihan\BMP_Process\" ; if
 g++ BMPNegatif.cpp -o BMPNegatif } ; if ($?) { .\BMPNegatif }
Input the file name : D:\\bubu.bmp
```

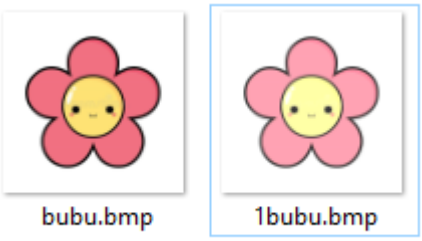
And for the output file, we will place it at the same place where the original file placed.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\LENOVO> cd "d:\KULIAH SMT 2\latihan\BMP_Process\" ; if ($?) {
 g++ BMPNegatif.cpp -o BMPNegatif } ; if ($?) { .\BMPNegatif }
Input the file name : D:\\bubu.bmp
The format file must bmp and located in the same disk with this program f
ile

The output file name : D:\\1bubu.bmp
increasing brightness [1-255] :  50
PS D:\KULIAH SMT 2\latihan\BMP_Process>
```



E. Conclusion

The function of this program is to make the image or the picture lighter than the original one. This program uses binary file handling to increase the brightness of the image by increasing the RGB value on its pixel according to the user input.