

Department of Computing and Information Technology
COMP 1602 Computer Programming II
2023/2024 Semester 2
Assignment 3

Date Due:

Sunday April 14, 2024 @ 11.55 pm

Overview

The *Graphics Interchange Format* (GIF) is a format for storing an image on a computer. A GIF file is a binary file where the bytes of an image are stored according to a specific layout. The GIF format allows multiple GIF images to be stored in one file. This is known as an *animated GIF* since Web browsers and other software can display the images in quick succession, creating an animation effect.

Structs for Image, AnimFrame, and Animation

A struct is used to store the bytes of an image. It is declared as follows:

```
struct Image {
    unsigned char descriptor [10];    // 10 bytes of data
    unsigned char colourTable [768]; // 256 x 3 bytes (red, green, blue)
    unsigned char data [10000000];    // to store image data
    int sizeData;                     // number of bytes in data array
};
```

Each image in an animation is referred to as a *frame*. To store a frame of an animation and the length of time the frame must be displayed, an *AnimFrame* struct is used. It is declared as follows:

```
struct AnimFrame {
    Image * image;           // address of the image to display
    int duration;            // how long to display the image
};
```

An animation consists of an array of *AnimFrames* (at most 20). It is declared as follows:

```
struct Animation {
    AnimFrame frames[20];    // array of frames
    int numFrames;           // number of actual frames stored
};
```

So, an animation consists of an array of *AnimFrames* where each *AnimFrame* contains the address of an *Image* struct and the length of time the image is displayed.

Special Byte Blocks

Several blocks of bytes are necessary when saving an *Image* struct or an *Animation* struct to a GIF file. These are shown below:

71	73	70	56	57	97
0	1	2	3	4	5

Header (6 bytes)

88	2	144	1	112	0	0
0	1	2	3	4	5	6

Logical Screen Descriptor (7 bytes)

33	255	11	78	69	84	83	67	65	80	69	50	46	48	3	1	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Application Extension (19 bytes)

33	249	4	8	100	0	0	0
0	1	2	3	4	5	6	7

Graphics Control Extension (8 bytes)

Functions to Write

Image * readImage (char filename[]);

This function reads the GIF image file using the filename supplied as a parameter and creates an *Image* struct in memory. It returns the address of the *Image* created. The following algorithm should be used to read the bytes into the different fields of the *Image* struct:

1. Ignore the first 13 bytes from the image file.
2. Read the next 10 bytes from the image file into the *descriptor* field.
3. Read $256 * 3$ (= 768) bytes from the image file into the *colourTable* field.
4. Read the remaining bytes from the image file into the *data* field. **Ignore** the last byte by reducing the number of bytes read by 1. Update the *sizeData* field so that it contains the number of bytes read in the *data* field.

Function	How to Implement Desired Effect
void toGrayScale (int * red, int * green, int * blue)	Find the average of the red, green, and blue values passed as parameters. Set the red, green, and blue parameters to the average.
void toBlueTint (int * red, int * green, int * blue)	Find the average of the red, green, and blue values passed as parameters. Set the red and green parameters to the average (blue is unchanged).
void toRedTint (int * red, int * green, int * blue)	Find the average of the red, green, and blue values passed as parameters. Set the green and blue parameters to the average (red is unchanged).
void toSepia (int * red, int * green, int * blue)	<p>Change the red, green, and blue parameters according to the following formula:</p> $\begin{aligned} \text{newRed} &= \text{red} * 0.393 + \text{green} * 0.769 + \text{blue} * 0.189 \\ \text{newGreen} &= \text{red} * 0.349 + \text{green} * 0.686 + \text{blue} * 0.168 \\ \text{newBlue} &= \text{red} * 0.272 + \text{green} * 0.534 + \text{blue} * 0.131 \end{aligned}$ <p>You must ensure that the new values for red, green, and blue don't go above 255.</p>
void brighten (int * red, int * green, int * blue, int percent)	<p>Increase the value of the red, green, and blue parameters by the percentage specified.</p> <p>You must ensure that the new values for red, green, and blue don't go above 255.</p> <p>When calling this function, increase the brightness by 60%.</p>

Animation * initAnimation (); // NB: This function is given

This function creates a new *Animation* without any frames. It returns the address of the *Animation* created.

void addFrame (Animation * animation, Image * image, int duration);

This function does two things. It first creates an *AnimFrame* based on the *image* and *duration* passed as parameters. The duration value must be between 1 and 255 (representing 0.01 to 2.55 seconds). The *AnimFrame* is then inserted at the end of the array of *AnimFrames*. The *numFrames* field of the animation must be updated accordingly.

void saveAnimation (Animation * animation, char filename[]);

This function saves the *Animation* to the file passed as a parameter. The algorithm to do so is as follows:

1. Write 6 bytes from **Header** to the image file.
2. Write 7 bytes from **Logical Screen Descriptor** to the image file.
3. Write 19 bytes from **Application Extension** to image file.
4. For each *AnimFrame* in the *Animation* {
 - a) Set **Byte 4** of the **Graphics Control Extension** to the *duration* of the *AnimFrame*.
 - b) Write 8 bytes from **Graphics Control Extension** to the image file.
 - c) Write 10 bytes from *descriptor* field of image to the image file.
 - d) Write 256*3 (=768) bytes from *colourTable* field of image to the image file.
 - e) Write *sizeData* bytes from *data* field of image to image file.}
5. Write a terminating byte (with value 59) to the image file.

Things to Do in *main* Function

In the *main* function, you must write code which uses the functions from the previous section to perform several operations involving GIFs such as applying effects to individual GIFs and creating animated GIFs. All you have to do is call the relevant functions to do what is required.

1. Create an *Image* for each of the following files: *Saskatoon.gif*, *MoraineLake.gif*, *Maracas.gif*, *ScarletIbis.gif*, and *PigeonPoint.gif*.
2. Apply the *grayScale* effect to *Saskatoon* and save the *Image* to *Saskatoon-GS.gif*.
3. Apply the *blueTint* effect to *MoraineLake* and save the *Image* to *MoraineLake-BT.gif*.
4. Apply the *sepia* effect to *Maracas* and save the *Image* to *Maracas-S.gif*.
5. Apply the *brighten* effect to *ScarletIbis* and save the *Image* to *ScarletIbis-B.gif*.
6. Apply the *redTint* effect to *PigeonPoint* and save the *Image* to *PigeonPoint-RT.gif*.
7. Create an empty *Animation* and call it *scenery*.
8. Add *Maracas* to *scenery*.
9. Add *MoraineLake* with the *blueTint* effect to *scenery*.
10. Add *PigeonPoint* to *scenery*.
11. Add *Maracas* with the *sepia* effect to *scenery*.
12. Add *Saskatoon* to *scenery*.
13. Save the *scenery Animation* to *Scenery-Animation.gif*.
14. Create an empty *Animation* and call it *random*.
15. Add 7 frames to *random* where the frames are randomly selected from the images in (1) to (6) above. The duration of each frame is randomly chosen from the range 50 to 255.
16. Save the *random Animation* to *Random-Animation.gif*.
17. Call the *showWebPage* function (provided in *Assignment3.cpp*) with "Assignment3.html" before returning from *main*. If you are using a non-Windows computer, this will not work, and you should comment this line in the code.

What You are Given

You are given the files listed in the following table:

Filename	Description
Assignment3.html	An HTML file which displays the GIFs created by your program on a Web browser.
Assignment.cpp	Contains a template for writing your code for Assignment 3. Use this as a starting point for your program. You can delete code already written in the functions and you can add your own functions.
Five GIF Files (Saskatoon.gif, MoraineLake.gif, Maracas.gif, ScarletIbis.gif, and PigeonPoint.gif)	These files are to be used for some of the operations in the <i>main</i> function.

What to Submit

Submit the code for the program in a single .cpp file (and nothing else).

It would make it much easier to mark your program if you can include the following information at the top of your program:

```
// Item 2: created GIF successfully? Yes/No
// Item 3: created GIF successfully? Yes/No
// Item 4: created GIF successfully? Yes/No
// Item 5: created GIF successfully? Yes/No
// Item 6: created GIF successfully? Yes/No
// Item 13: created animated GIF of Scenery successfully? Yes/No
// Item 16: created animated GIF of random images successfully? Yes/No
// Item 16: Web page launches successfully from program? Yes/No
// Any other comments?
```

Programming Notes

1. Binary file formats are very rigid in their layouts. One byte that is out of place can corrupt the whole file. If your program has created the required output files but they are not being displayed correctly, most likely there were errors in writing the bytes to the files. You need to ensure that **every** byte is inserted in the right place.
2. To copy a string literal (e.g., "Maracas.gif") to a C-string, *filename*, use the following code:

```
char filename [25];
strcpy (filename, "Maracas.gif");
```

3. To open a binary file where the filename is stored in a C-string, *filename*, use the following code:

```
inputFile.open(filename, ios::binary | ios::in);
```

4. Use the following approach to achieve Part (4) of the *readImage* algorithm:



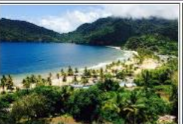








```

Read one byte from image file into aByte
While not end-of-file {
    Store aByte in the relevant location of the data array
    Update the next available location of the data array
    Read one byte from image file into aByte
}
Reduce the number of bytes read by 1

```

Sample Output

COMP 1602 Assignment 3

GIFs for Items 1, 2, 3, 4, 5, and 6					
Original Images:					
Images with Effects:					
Animated GIFs					
Scenery Animation (Items 7, 8, 9, 10, 11, 12, and 13)			Random Animation (Items 14, 15, and 16)	