This is CS50

CS50's Introduction to Computer Science

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

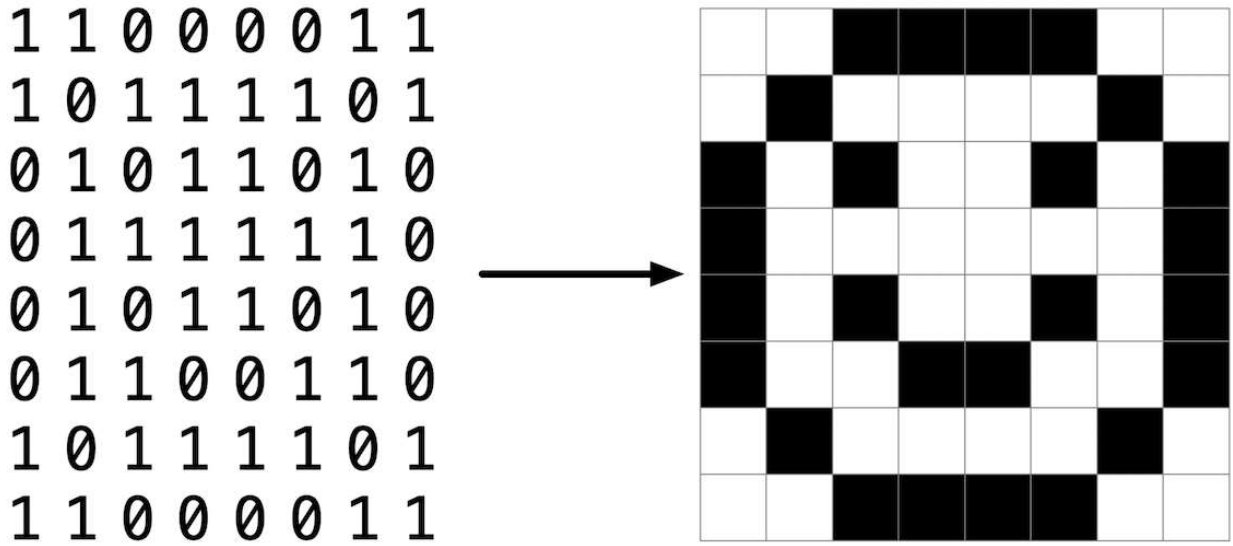
David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Lab 4: Smiley

Learning Goals

- Learn how to work with images
- Practice manipulating pixels

Background



You've seen in lecture a bit about how images are stored on a computer. In this lab, you'll practice working with a BMP file, actually the smiley face pictured here, and change all the black pixels to a color of your choosing.

However, the smiley face you'll be working with is not just made of of 0's and 1's, or black and white pixels, but consists of 24 bits per pixel. It uses eight bits to represent red values, eight bits for green and eight bits for blue. Since each color uses eight bits or one byte, we can use a number in the range of 0 to 255 to represent its color value. In hexadecimal, this is represented by `0x00` to `0xff`. By mixing together these red, green and blue values, we can create millions of possible colors.

If you look at `bmp.h`, one of the the helper files in the distribution code, you'll see how each `RGB triple` is represented by a `struct` like:

```
typedef struct
{
    BYTE rgbtBlue;
    BYTE rgbtGreen;
    BYTE rgbtRed;
}
RGBTRIPLE;
```

where `BYTE` is defined as an 8-bit integer.

You'll notice several files provided in the distribution code to handle the reading and writing of an image file, as well as handling the image's metadata or "headers". You'll be completing the function `colorize` in `helpers.c`, which already has as input parameters, the image's height, width, and a two-dimensional array of `RGBTRIPLE`'s which create the image itself.

⊞ Hints

Demo

Recorded with asciinema

Getting Started

Open VS Code (https://cs50.dev/).

Start by clicking inside your terminal window, then execute `cd` by itself. You should find that its "prompt" resembles the below.

```
$
```

Click inside of that terminal window and then execute

```
wget https://cdn.cs50.net/2022/fall/labs/4/smiley.zip
```

followed by Enter in order to download a ZIP called `smiley.zip` in your codespace. Take care not to overlook the space between `wget` and the following URL, or any other character for that

matter!

Now execute

```
unzip smiley.zip
```

to create a folder called `smiley`. You no longer need the ZIP file, so you can execute

```
rm smiley.zip
```

and respond with "y" followed by Enter at the prompt to remove the ZIP file you downloaded.

Now type

```
cd smiley
```

followed by Enter to move yourself into (i.e., open) that directory. Your prompt should now resemble the below.

```
smiley/ $
```

If all was successful, you should execute

```
ls
```

and you should see `bmp.h`, `colorize.c`, `helpers.c`, `helpers.h`, `Makefile`, and `smiley.bmp`.

If you run into any trouble, follow these same steps again and see if you can determine where you went wrong!

Implementation Details

Open up `helpers.c` and notice that the `colorize` function is incomplete. Note that the image's height, width and a two-dimensional array of pixels is set up as the input parameters for this function. You are to implement this function to change all the black pixels in the image to a color of your choosing.

You can compile your code by simply typing `make` at the `$` prompt.

You then execute the program by typing:

```
./colorize smiley.bmp outfile.bmp
```

where `outfile.bmp` is the name of the new bmp you are creating.

Thought Question

- How do you think you represent a black pixel when using a 24-bit color BMP file?
- Is this the same or different when mixing paints to repesent various colors?

How to Test Your Code

Your program should behave per the examples below.

```
smiley/ $ ./colorize smiley.bmp smiley_out.bmp
```

When your program is working correctly, you should see a new file, `smiley_out.bmp` in your `smiley` directory. Open it up and see if the black pixels are now the color you've specified.

You can check your code using `check50`, a program that CS50 will use to test your code when you submit, by typing in the following at the `$` prompt. But be sure to test it yourself as well!

```
check50 cs50/labs/2023/x/smiley
```

To evaluate that the style of your code (indentations and spacing) is correct, type in the following at the `$` prompt.

```
style50 helpers.c
```

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/labs/2023/x/smiley
```

▶ **Want to see the staff's solution?**