Bryce Jarboe
RedID 825033151
9/15/2023
CS420

Imperative Programming Report (C)

**Progression:**

The development process for this project followed a systematic approach. Initially, I began by outlining the step-wise logic in comments, describing the intended functionality of each function. Since most of this logic was directly derived from the assignment prompt, the transition to actual implemented statements was quite easy, and proceeded relatively quickly. The main challenge I encountered during development was troubleshooting and debugging. To my surprise, the majority of the core logic was correct, but I encountered minor syntax errors and issues related to the positioning of the file pointer, as well as general difficulties using the stdio file reading/writing functions. These errors led to numerous segmentation faults, which were especially difficult since the debugger provided no immediate solutions. Additionally, the image file became corrupted at some point, resulting in incorrect outputs despite having the correct code in place.

**Read Function:**

The read function plays a crucial role in the program by opening the specified image file and creating a file pointer for its following operations. It relies on a series of fscanf() statements and a fgetc() statement to parse the *relevant* information from the PGM file. The fscanf() statements are responsible for extracting data such as image width, height, and maximum pixel value. The primary focus of this function is on retrieving the width and height values, which are essential for allocating memory for the 2D matrix of PixelGray structures that will store the image data.

**Write Function:**

The write function takes an existing 2D matrix of PixelGray structures and saves the data to a new file. The function initiates a new file pointer and uses fprintf to write the necessary numeric data into the PGM header, including the width and height of the image. Following the header information, the function iterates through the entire matrix, using fwrite to write each element into the new file.

**Threshold Function:**

The threshold function is responsible for processing the input matrix, identifying pixels with values greater than a threshold (80), and skewing the value towards its maximum or minimum appropriately. The function initializes a new matrix, matrixC, to store the thresholded image data, allocating memory

the same way as the read function and ensuring that it has the same dimensions as the original matrix. It then traverses the original matrix, comparing each pixel's value with the threshold. This process creates a high-contrast representation of the image.

*1 Threshold*

**Rotate Function:**

The rotate function is responsible for rotating the input matrix by 90 degrees counterclockwise. It achieves this by creating a new transposed matrix, matrixT, and swapping the row and column indices of the original matrix's elements. Similar to the other functions, the rotate function first allocates memory for the new matrixT to ensure it has the appropriate dimensions for the rotated image. It then iterates through the original matrix, transferring the pixel values to the transposed matrix with swapped row and column indices. This results in a rotated image suitable for further processing or saving. The key word here is *transpose*, as the image is not technically *rotated*. Due to the mathematical rigidity of the transposing function, calling the rotation again would return the original image.

*2 rotate | 3 rotate_again*