**Image Converter Coursework Report**

**Introduction:**

The aim of this coursework was to create an Image Converter. The program is meant to take in a .bif file which would contain a colour image and then output a .pgm file, which would contain the same image but in greyscale. The program also has the option for the user to indicate whether or not they want the image to be displayed or scaled.

**The Program:**

One of the first things that I did was create input and output files for the program. Once created, I made the input file NULL so that it was completely empty and ready to be written to. The files were then opened so that they could be accessed. The input in reading mode and the output in writing mode.

|  |
| --- |
| fopen\_s(&InputFile,InputName,"r"); |
| fopen\_s(&OutputFile,OutputName,"w"); |
|  |  |

DOES THE OUTPUT FILE NEED TO BE NULLTOO -\_\_\_-?

The actual input image data was expressed as an integer triplet which represented an RGB value. I decided to use a struct to store the information regarding the pixel input data so that it would all be grouped together and also it would be easier to access specific parts of the data. The struct that I created is shown below. I assigned them all to the variable type “Pixel”.

|  |
| --- |
| typedef struct Pixel { |
|  | int R; |
|  | int G; |
|  | int B; |
|  | int Intensity; |
|  | }Pixel; |

As the number of input arguments to the program are not always going to be exactly the same, it was important for them to be assigned to the correct variables. This was implemented by using an else if statement which assigned the correct inputs to the correct variable names as can be seen by the flowchart in Figure 1. The minimum number of inputs is three - the executable name, the input image file name, the output image file name - and the maximum number is 5 – the above plus an option to display the image and an option to scale the image -.

After the input and output files are opened, the program figures out how many rows and columns of data it has been given by the input file. It does this by using the ReadNumber function to first read the first integer from the input file and then assigning this to the variable nRows. It then does the same again but this time reading the second number and assigning it to the variable nColumns. These variables are then used later by the program to know how much memory needs to be allocated for the pixels to be stored.

Since the output file is required to contain the line of text that is given in the input file describing the image being converted, the ReadInfo function is called to read in the description line and assign it to the variable DescriptionLine. This means that it can be easily be written onto the output file.

* Uses ReadInfo function to find out what the description Line says

As the format for the output file is always the same, it was possible to just write these on to the file using the main part of the program and not creating separate functions. These were the magic number: “P2”, the description line from the input file, the number of rows and columns of data and the max intensity. These were all written on to the output file using the fputs function. Due to the max intensity always being 255, I created a define variable after the header files, where I assigned 255 to the variable MaxIntensity.

* **Should the 2D array be rows\* colums\* sizeof Pixel?? ☹**

To create a 2D array in which to store the pixels I used the malloc function to allocate enough memory. The amount of memory is equal to the number of rows multiplied by the size of a Pixel.

Pixel \*\*PixelArray = (Pixel \*\*)malloc(nRows \* sizeof(Pixel \*));

* Iterates through the code to save all of the pixels into the 2d array
* Conversion
* Iterates through again this time writing the data to the output file
* **ADD SPACE BETWEEN EVERY ITERATION**
* **DO I NEED TO CLOSE THE FILE????Apparently it’s necessary**

Fig.1.Input Arguments Flowchart

YES

YES

Are there 5 input arguments?

NO

ERROR MESSAGE

Are there less than 3 or more than 5 arguments?



YES

Set the display and Scale flag. Assign third argument to Input Name and fourth argument to Output Name

NO

YES

NO

YES

Set the Scale flag.

Set the Display flag.

Is the second character of the first argument “d”?

Assign the second argument to Input Name and the third to Output Name

Are there 4 input arguments?

Are there 3 input arguments?

NO

Assign first argument to Input Name and the second to Output Name.

**Functions:**

Function 1: ReadNumber

**DECLERATIONNNNNNNN**

The aim of this function is to read in the information from the data file as a string and then convert the string to an integer. In this program it is used to read in the first line of information from the input file which contains the number of rows and columns needed to contain all of the image information. It uses the function fscanf\_s to read in the string as this function stops when it reaches a whitespace and the first line of the input file format is the number of rows and then the number of columns separated by a whitespace. The function takes in the information as a string and then uses strtol to convert the string to an integer, which is then returned by the function. The input parameter of this function the name of the file that the string is being read from. The function is called in the main as follows:

nRows = ReadNumber(InputFile);

nColumns = ReadNumber(InputFile);

Function 2: ReadInfo

* Declaration
* Description

The aim of this function is to read into the program the line of text provided by the input file describing the image being converted. This restricts the description line size to 255 including the string terminator. The function works by using a for loop that increments until it has reached the 255th iteration or the string terminator.

Function 3: ReadPixel

* Declaration

The aim of this function is to read one RGB triplet and convert it to a greyscale value, known by the program as the variable pix.Intensity . The way in which this is done is by averaging the RGB values. This function calls upon the first function ReadNumber in order to read the RGB values from the input text file and then simple arithmetic is used to average them. This then results in the intensity being returned as a pixel. This function is called whilst the program is iterating through the 2D pixel array for the first time. It means that every pix.Intensity value can be stored in the 2D array. OUTPUT A PIXEL? IF I ADD IN A CHECK THEN TALK ABOUT THAT

**IS THIS ACTUALLY USED THO??**

Function 4: WriteOutputPixel

* Declaration
* Description

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Error Tests:**

* Move PrintF on number of arguments

The ReadNumber function needs to convert a string to an integer. If this conversion does not occur, it will mean that the program will not be able to continue as the integer outputted is crucial to everything else running. The way in which this test runs is by comparing the start address of the string to the end address of the string. If they are both the same, it means that the pointer hasn’t moved and the conversion hasn’t happened. But if they are different the pointer has moved and a conversion has been carried out so the program can go on. This was implemented using an if else statement.

* NEED TO ADD THE IF THING FOR LESS THAN 3 AND MORE THAN 5

\*\*IF I DON’T ADD THIS DELETE THE DESCRIPTION\*\*

As the number of input parameters to the program is variable, this error test checks whether the number of input arguments is valid. As there is a minimum of three and a maximum of five, if there is a number of input arguments not within this range, it will display an error message to inform the user that the number of input arguments they have entered isn’t valid.

* NEED TO ADD IF PIX INTENSITY IS BIGGER THAN 255 DISPLAY ERROR MESSAGE
* ADD IN ERROR FOR IF THE FILE NAME THAT IS INPUTTED DOESN’T EXIST,

*If file== null ewcoding*

**Testing Errors:**

* Test file with incorrect number of input arguments
* Test the conversion
* Test pix.Intensity
* Test incorrect file name

**Improvements:**

A potential improvement to the code could be to have it to include a function to work out the maximum intensity of the RGB values that were provided by the input file. At the moment it has a variable defined right after the header to say that it will always be 255, which is the maximum it could ever be but every input file would have its own maximum intensity.