

## image.h

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
// define bmp header size , colorTable and custom image size as per the structure of bmp file.
```

```
#define BMP_HEADER_SIZE 54
```

```
#define BMP_COLOR_TABLE_SIZE 1024
```

```
#define CUSTOM_IMG_SIZE 1024 * 1024
```

```
int imageReader(const char *imgName, int *_height, int *_width, int *_bitDepth, unsigned char  
*_header, unsigned char *_colorTable, unsigned char *_buffer);
```

```
int imageWriter(const char *imgName, unsigned char *header, unsigned char *colorTable, unsigned  
char *buffer, int bitDepth);
```

```
int initialize(const char *read_image, const char *write_image);
```

```
int initialize(const char *read_image, const char *write_image)
```

```
{
```

```
    // Initialize datatypes of header, height, width, BitDepth to use it after reading.
```

```
    int imgWidth, imgHeight, imgBitDepth;
```

```
    unsigned char imgHeader[BMP_HEADER_SIZE];
```

```
    unsigned char imgColorTable[BMP_COLOR_TABLE_SIZE];
```

```
    unsigned char imgBuffer[CUSTOM_IMG_SIZE];
```

```
    // Call the read and write function
```

```
    int checkReader = imageReader(read_image, &imgWidth, &imgHeight, &imgBitDepth,  
&imgHeader[0], &imgColorTable[0], &imgBuffer[0]);
```

```
    if (checkReader == 0)
```

```
    {
```

```
        printf("Read Successful");
```

```
    }
```

```
    else
```

```
    {
```

```

        printf("Read Unsuccessful");
    }

    int checkWriter = imageWriter(write_image, imgHeader, imgColorTable, imgBuffer, imgBitDepth);

    if (checkWriter == 0)
    {
        printf("\nWrite Successful");
    }
    else
    {
        printf("\nWrite Unsuccessful");
    }

    return 0;
}

```

```

int imageReader(const char *imgName, int *_height, int *_width, int *_bitDepth, unsigned char
*_header, unsigned char *_colorTable, unsigned char *_buffer)
{
    int i;

    // Initialize a FILE pointer for reading
    FILE *streamIn;

    // Open the file to read
    streamIn = fopen(imgName, "rb");

    // Check if the FILE pointer is able to access
    if (streamIn == (FILE *)0)
    {
        printf("Unable to read file\n");
    }

    for (i = 0; i < 54; i++)
    {
        // Read the header.

```

```

        _header[i] = getc(streamIn);
    }

    // Read the width, height, bitDepth from header.
    *_width = *(int *)&_header[18];
    *_height = *(int *)&_header[22];
    *_bitDepth = *(int *)&_header[28];

    // Check if the bitDepth is less than 8 and if it is read the colortable from streamIn.
    if (*_bitDepth <= 8)
    {
        // Read the colortable of size unsigned char from streamIn. 1024 being the
        // no of elements of size mentioned before.
        fread(_colorTable, sizeof(unsigned char), 1024, streamIn);
    }
    else
    {
        printf("BitDepth is greater than 8, can't read the file");
        return 1;
    }

    // Read the data (buffer) from the streamIn.
    fread(_buffer, sizeof(unsigned char), CUSTOM_IMG_SIZE, streamIn);

    // Close the FILE pointer.
    fclose(streamIn);

    return 0;
}

int imageWriter(const char *imgName, unsigned char *header, unsigned char *colorTable, unsigned
char *buffer, int bitDepth)
{
    // Open the file for write.
    FILE *FO = fopen(imgName, "wb");

    // Write the header of size 54 bytes

```

```
fwrite(header, sizeof(unsigned char), 54, FO);  
  
// Check to see if the bitDepth is less than 8.  
if (bitDepth <= 8)  
{  
    // Write the colortable of size unsigned char from streamIn. 1024 being the  
    // no of elements of size mentioned before.  
    fwrite(colorTable, sizeof(unsigned char), 1024, FO);  
}  
else  
{  
    printf("BitDepth is greater than 8, can't write the file");  
    return 1;  
}  
  
// Write the data.  
fwrite(buffer, sizeof(unsigned char), CUSTOM_IMG_SIZE, FO);  
  
// Close the write file pointer.  
fclose(FO);  
  
return 0;  
}
```

## Main function

### main.c

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include "image.h"
```

```
int main()
```

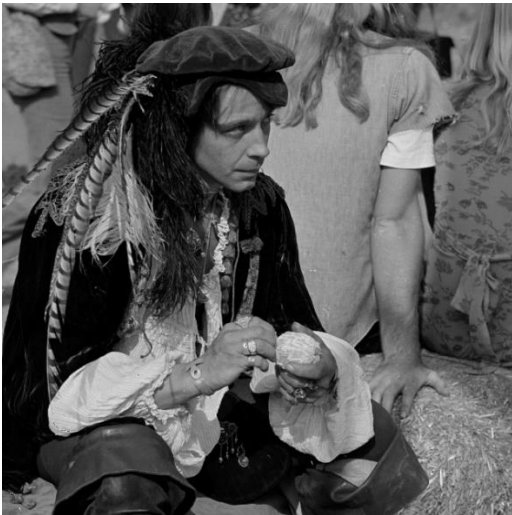
```
{
```

```
    // call initialize function and give the arguments of bmp file to read and write.
```

```
    initialize("image.bmp", "image_copy.bmp");
```

```
}
```

**image.bmp**



**image\_copy.bmp**

