

Forum:[Hack Forums](#)

Thread: [Reading/Writing .bmp file](#)

nuvexy. - October 2, 2019, 8:26 pm

Can anyone help me out with this lab for school? I'm lost.

Here are the lab directions:

1. Download the test files: lab.bmp (3 MB) and ut.bmp (112 KB). You will write all of your code in C++.
2. In this step, you will check to make sure you understand how to read a bitmap file. Write a program whose first command line argument is the input bitmap file and the second command line argument is the output bitmap file.
3. Create a structure called "Pixel" that stores the pixel color channels. [See bitmap file structure for details on how to construct a pixel]
4. Pixel, BitmapFileHeader, and BitmapInfoHeader are the ONLY structures you may use in this lab. Pixel must have a size of 3 bytes.
5. For the input file, read the offset (from file header), width, and height (from info header) from the file by seeking to the appropriate location.
6. Create a pixel array on the heap to store all of the pixels in memory. You must use a pointer of data type Pixel *.
7. Read in all of the pixels into memory using your pixel pointer. DO NOT FORGET to skip the padding after each row. This means you must read row-by-row rather than the entire pixel array all at once. You must use file seeking to skip the padding and move the file location to the next row. You must determine the amount of padding using the row size.
8. Now, using an output file, write the exact same file and info header to the output file.
9. Write the pixel data. Do not forget to include the extra padding after each row (if applicable).

Verifying:

1. You must have a Pixel structure and use it for your pixel array. You must have a BitmapFileHeader structure and a BitmapInfoHeader structure as well.
2. You must allocate using the heap using the new and delete operators. Do not use malloc or free or vectors.

Here's what I have so far:

```
Code #include <cstdio>
#include <cstdlib>
```

```
#pragma pack(push, 2)
struct BitmapFileHeader
{
```

```

uint16_t type;
uint32_t size;
uint16_t reserved_1;
uint16_t reserved_2;
uint32_t offset;
};

```

```

struct BitInfoHeader
{
    uint32_t size;
    uint32_t width;
    uint32_t height;
    uint16_t planes;
    uint16_t bitcount;
    uint32_t compression;
    uint32_t imagesize;
    uint32_t x_pixels_per_meter;
    uint32_t y_pixels_per_meter;
    uint32_t color_used;
    uint32_t color_important;
};

```

```

struct pixel {
    uint8_t blue;
    uint8_t green;
    uint8_t red;
};

```

```

#pragma pack(pop)

```

```

int main()
{
    FILE*fin;
    BitmapFileHeader bfh;
    fin = fopen(input, "rb");

    if (nullptr == fin) {
        perror("lab.bmp");
        return -1;
    }

    if(sizeof(BitmapFileHeader) != fread(&bfh, 1, sizeof(bfh), fin)) {
        printf("Unable to read bitmap file header.\n");
        return -2;
    }

    printf("Size of file header = %lu\n", sizeof(BitmapFileHeader));
    int8_t first = (bfh.type >> 8) & 0xff;
    int8_t second = bfh.type & 0xff;
    printf("File type = %c%c\n", first, second);
    printf("File Size = %u\n", bfh.size);
}

```

```

printf("File offset = %u\n", bfh.offset);
fseek(fin, bfh.offset, SEEK_SET);
pixel p;
fread(&p, 1, sizeof(p), fin);
printf("R = %u, G = %u, B = %u\n", p.red, p.green, p.blue);
return 0;
}

```

HFDone - October 3, 2019, 2:56 am

<https://solarianprogrammer.com/2018/11/19/cpp-reading-writing-bmp-images>

YouthInAsia - October 3, 2019, 4:15 am

It says to use command line arguments.

```

Code int main(int argc, char* argv[]){
char* input = argv[1];
char* output = argv[2];

}

```

nuvexy. - October 3, 2019, 7:42 am

(Oct 3, 2019 04:15 AM)YouthInAsia Wrote: [\[CITE\]](#) It says to use command line arguments.

```

Code int main(int argc, char* argv[]){
char* input = argv[1];
char* output = argv[2];

```

```

}

```

That's one of my problems. I can't use a char cause I have to use fseek and fread and those require a FILE pointer.

HFDone - October 3, 2019, 9:32 am

[fopen - C++ Reference](#)

```

FILE fopen ( const char filename, const char * mode );
fin = fopen(argv[1], "rb");

```

nuvexy. - October 3, 2019, 10:23 am

(Oct 3, 2019 09:32 AM)HFDone Wrote: [\[CITE\]](#) [fopen - C++ Reference](#)

```

FILE fopen ( const char filename, const char * mode );
fin = fopen(argv[1], "rb");

```

Alright great! That takes care of the command line argument trouble! Thank you!

Now how can I store the pixels in an array?

0xDEAD10CC - October 3, 2019, 1:58 pm

(Oct 3, 2019 10:23 AM)nuvexy. Wrote: [\[CITE\]](#) Alright great! That takes care of the command line argument trouble! Thank you!

Now how can I store the pixels in an array?

Use an unsigned char array for the bytes. Did they teach you anything in class or were you not paying attention?

nuvexy. - October 3, 2019, 2:02 pm

(Oct 3, 2019 01:58 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Use an unsigned char array for the bytes. Did they teach you anything in class or were you not paying attention?

Just transferred from a community college that taught Java with a really poor curriculum. Thrown into this class that uses C++ and C not knowing virtually anything taught before. Having a bit of

trouble. Seeing as this is lab # 5, I'd say that seeking help for 1/5 of the labs so far isn't too bad of a ratio.

Thanks for the concern tho, asshole.

YouthInAsia - October 3, 2019, 2:19 pm

(Oct 3, 2019 07:42 AM)nuvexy. Wrote: [\[CITE\]](#) That's one of my problems. I can't use a char cause I have to use fseek and fread and those require a FILE pointer.

You dumb? You're supposed to call fopen using those two file names passed in command line arguments

[warn]

Number23 - October 3, 2019, 7:04 pm

(Oct 3, 2019 02:19 PM)YouthInAsia Wrote: [\[CITE\]](#) You dumb? You're supposed to call fopen using those two file names passed in command line arguments

All the functions should take the inout files as a command, ./ .txt .txt...

YouthInAsia - October 3, 2019, 7:54 pm

(Oct 3, 2019 07:04 PM)xllVllaGiiKz Wrote: [\[CITE\]](#) All the functions should take the inout files as a command, ./ .txt .txt...

Yes..what do you think my code did?

(Oct 3, 2019 10:23 AM)nuvexy. Wrote: [\[CITE\]](#) Alright great! That takes care of the command line argument trouble! Thank you!

Now how can I store the pixels in an array?

You serious ? I gave you that solution were you too dumb to see it?

[warn]

int main() - October 4, 2019, 2:49 am

(Oct 3, 2019 07:54 PM)YouthInAsia Wrote: [\[CITE\]](#) Yes..what do you think my code did?

You serious ? I gave you that solution were you too dumb to see it?

why in the actual fuck are you this fucking mean. If you're too bothered to help then don't, dont be bringin your negative ass into this young mans life you bum.

0xDEAD10CC - October 4, 2019, 10:23 am

(Oct 3, 2019 02:02 PM)nuvexy. Wrote: [\[CITE\]](#) Just transferred from a community college that taught Java with a really poor curriculum. Thrown into this class that uses C++ and C not knowing virtually anything taught before. Having a bit of trouble. Seeing as this is lab # 5, I'd say that seeking help for 1/5 of the labs so far isn't too bad of a ratio.

Thanks for the concern tho, asshole.

You're asking for spoonfeed; you posed no real question for the community on what part you needed help with, you just posted what you completed so far (which is barely anything) and your assignment instructions itself. If you can't do the course then don't do it or ask the appropriate channel(s) for help (like your teacher/prof). Do you think you're special and entitled to a free walk-through because of your situation? ?

If you don't even understand the basic datatypes how are you ever going to understand the actual assignment itself and code given to you for dealing with file I/O, endianness, and everything else that comes from reading data and structures from a file? -- Thus, helping you is a waste of time if you cannot comprehend the provided solution.

What you're doing is cheating, and I'm not going to help for that. If you truly don't understand the entire assignment and not just parts of it, then that is something people bring up with their teacher/prof instead of cheating to get the marks they don't deserve.

themayonnaise - October 6, 2019, 3:13 pm

(Oct 2, 2019 08:26 PM)nuvexy. Wrote: [\[CITE\]](#) stuff

You're on the right track, the next step for you to do is to read the DIB header as you'll need this to calculate how much memory you need to allocate for the pixel array. This is actually quite easy, as its pretty much the same code you used to read in the file header. So just to make it clear, there are 3 fundamental parts of a bitmap which you need to read:

- Firstly, The file header. The file header holds information about the file itself -- first the magic number to identify that yes, this is a bitmap. Following this you can read in the file size and more importantly, the offset to the pixel array start. So far, you've read this in (and done a decent job!), and you're about to read in the pixel data.

- The next bit is called the DIB header. The DIB header doesn't hold information about the file, but rather the contents of the bitmap. So this is a bit more useful to us -- using the DIB header we can find the image width and height in pixels (and therefore, the total number of pixels we need to allocate memory for). The DIB header also has a bunch of other stuff, but for the purpose of this we don't really care. For example, we're not gonna read in weird compressed BI_CMYKKRLE8 or BI_JPEG bitmaps. We already know each pixel is only going to be 3 bytes (24 bits per pixel) and in BI_RGB format. After this, there is an optional color table section, followed by an optional gap. For our purposes we couldn't give a toss about this so don't worry about this section.

- Then, there's the pixel data! The pixel data section is a bit odd, because pixels are stored row-by-row rather than being in a contiguous block of memory. What's more weird is that number of bytes in each row should add up to a multiple of 4, so there might be some padding we need to skip. Otherwise, we might read in a bunch of zeros as the red channel of a new pixel, and fundamentally screw up our read process. It's fairly straightforward to calculate this though. After the pixel data there may be some other optional stuff but we don't really care about it so don't worry.

(A really good breakdown of the BMP file format can be found on the wiki page)

Reading in the DIB header

So to start off, the one bit you've missed is reading in the DIB file header into your BitInfoHeader. This should be done after you've read in the file header. Copying and modifying your code a bit, we can use the following to do this:

PHP Code

```
<?php
BitInfoHeader dib;

if (sizeof(BitInfoHeader) != fread(&dib, 1, sizeof(dib), fin))
{
    printf("Unable to read DIB header.\n");
    return -3;
}
```

And it's that easy! Now you have read the DIB header, you could do some optional error checking to make sure you're not about to read a weird format bitmap which your code doesn't support. For example, you could check if the size of the BitInfoHeader struct is different from the size of the DIB header (dib.size), because this might end up blowing your program up. You could also make sure that your bit-per-pixel count is 24 and the compression format is definitely BI_RGB (0).

Allocating space for the pixels

Now you have the width/height of the image in pixels, it's easy to allocate space for the pixels with:

PHP Code

```
<?php
//Calculate the number of pixels we need in the image, and allocate an
//array of structs this large
unsigned int pixelCount = (dib.width * dib.height);

//Allocate space pixel_count pixels for when we read in
pixel* pixels = new pixel[pixelCount];

//...

//Don't forget to call this before your program exits:
delete[] pixels;
```

Reading in the pixels

As your instructions mention, reading in bitmap pixel data is a bit of a ballache because you have to do it row-by-row and calculate the amount of padding you need to skip. To calculate the padding, you just need to find the next multiple of 4 and use the row size in bytes:

PHP Code

```
<?php
//Get the row size (width * 3 bytes), padding is just how many more
//bytes we need until its a multiple of 4
unsigned int rowSize = dib.width * 3;
unsigned int rowPadding = rowSize % 4;
```

Once these are defined, it's mega easy to actually read in the pixel data. You have the number of rows (dib.height), and the number of elements to read (dib.width), the number of bytes to read (dib.width * sizeof(pixel)), along with how many bytes to seek forward and skip (rowPadding). So, you could use this to successively read pixel data row-by-row into the array:

PHP Code

```
<?php
for (int row = 0; row < dib.height; row++)
{
    //So for each row, read rowSize bytes and then skip rowPadding bytes
    //..
```

```
//Where in the array are we going to read to? Well this is at the start of a row
//so we're at pixel (w * row)
int index = dib.width * row;

//Read it into the array, starting at this index, for a total of sizeof(pixel) * w bytes
fread(pixels + index, sizeof(pixel), dib.width, fin);

//Skip however many bytes of padding
fseek(fin, rowPadding, SEEK_CUR);
}
```

After this you should have all the pixels read into memory! I'll leave the task of writing it out to another file to you. That's pretty easy though, you just fopen a file with "wb", fwrite your file header, fwrite the DIB header, fseek to where the pixels should be written, fwrite them row by row with padding similar to the for loop above, and fclose.

Also take note that my code may be jank as fuck, so take it as inspiration for a better solution. Hopefully this has given you some pointers (get it?) and it helps you out. If you got any questions just let me know, as I can appreciate that learning C is a bit tough at times. I'm around on discord too! :)