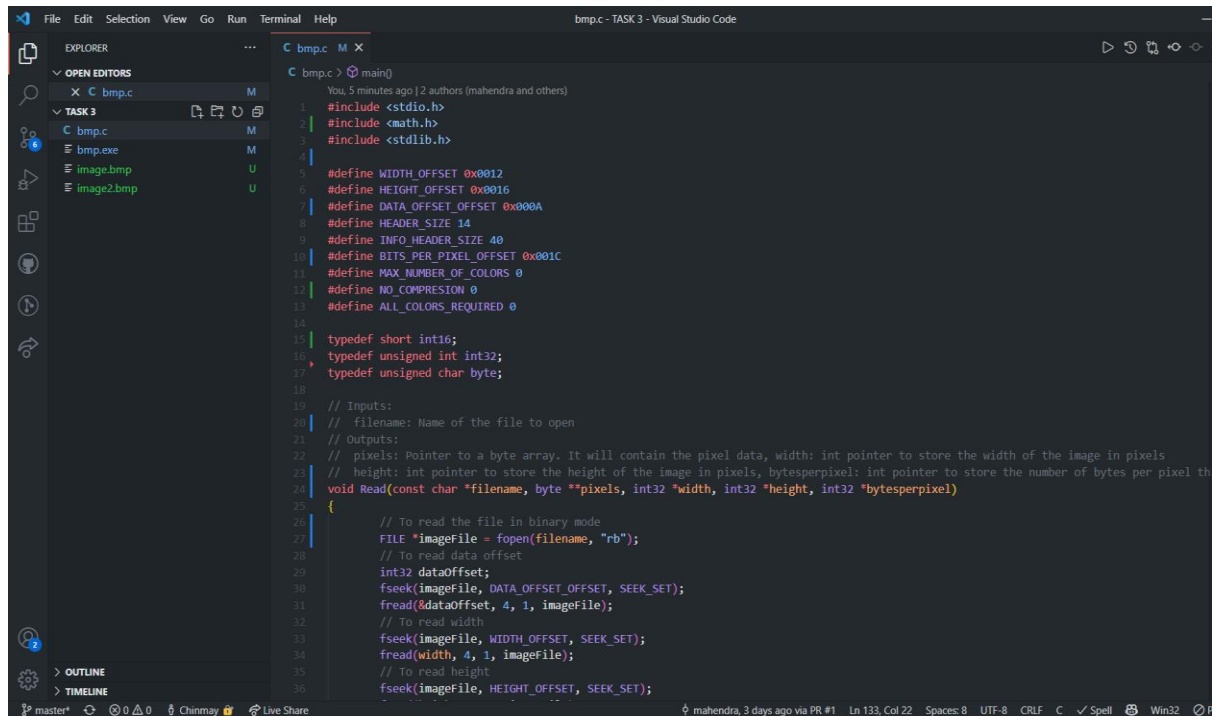


Task #3 Implement an Image File Reader and Writer

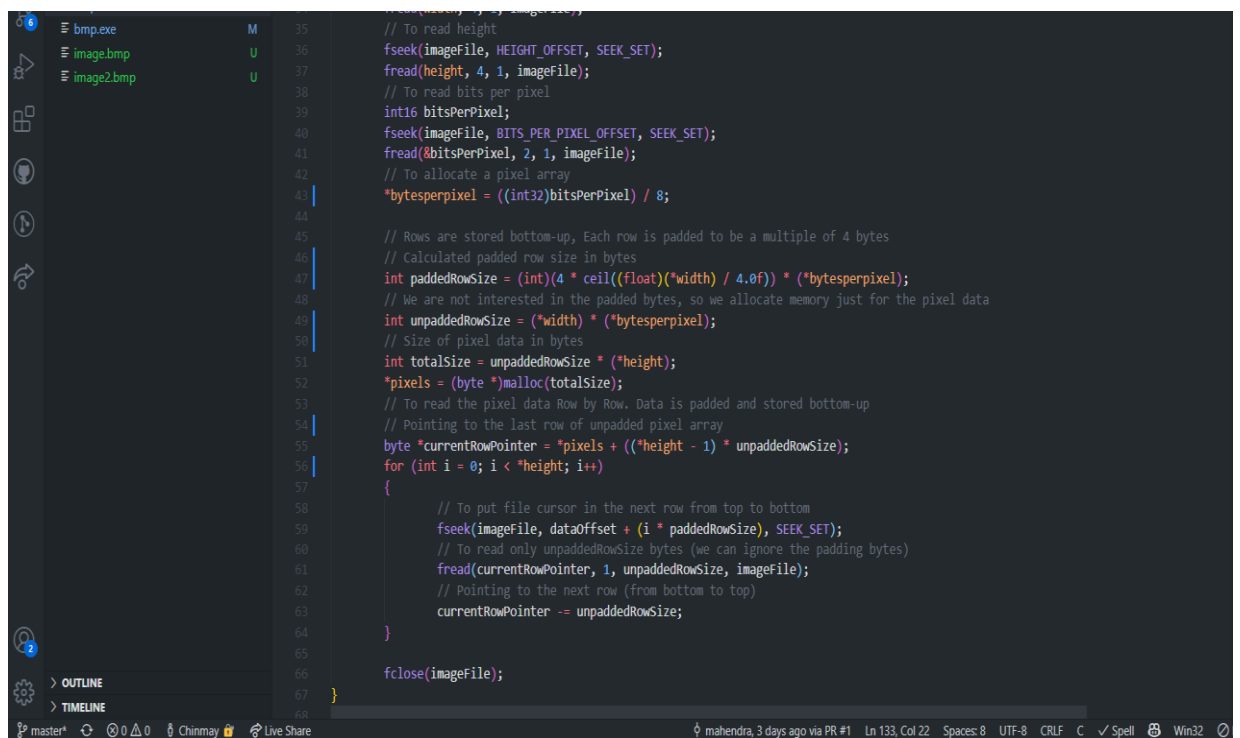
1. Function to Read Bitmap Image



```
File Edit Selection View Go Run Terminal Help
bmp.c - TASK 3 - Visual Studio Code

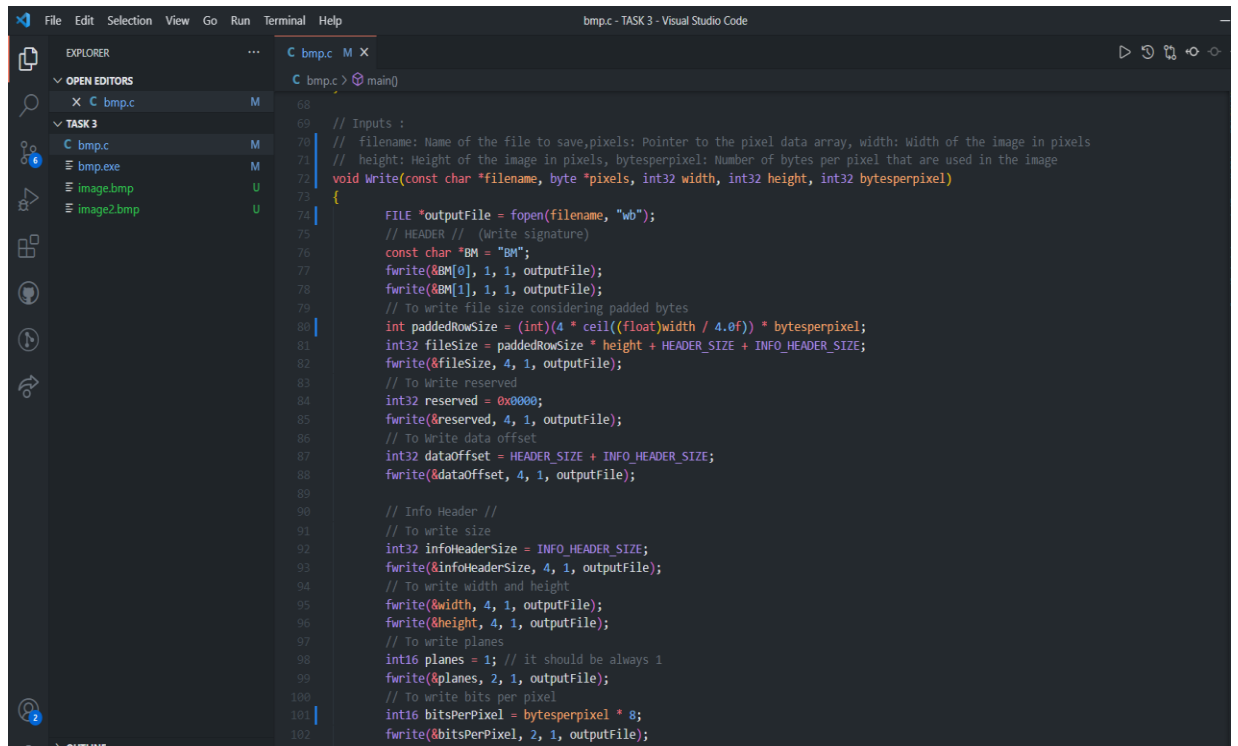
EXPLORER
  OPEN EDITORS
    C bmp.c
  TASK 3
    C bmp.c
    bmp.exe
    image.bmp
    image2.bmp

C bmp.c
1  You, 5 minutes ago | 2 authors (mahendra and others)
2  #include <stdio.h>
3  #include <math.h>
4  #include <stdlib.h>
5
6  #define WIDTH_OFFSET 0x0012
7  #define HEIGHT_OFFSET 0x0016
8  #define DATA_OFFSET_OFFSET 0x000A
9  #define HEADER_SIZE 14
10 #define INFO_HEADER_SIZE 40
11 #define BITS_PER_PIXEL_OFFSET 0x001C
12 #define MAX_NUMBER_OF_COLORS 0
13 #define NO_COMPRESSION 0
14 #define ALL_COLORS_REQUIRED 0
15
16 typedef short int16;
17 typedef unsigned int int32;
18 typedef unsigned char byte;
19
20 // Inputs:
21 // filename: Name of the file to open
22 // pixels: Pointer to a byte array. It will contain the pixel data, width: int pointer to store the width of the image in pixels
23 // height: int pointer to store the height of the image in pixels, bytesperpixel: int pointer to store the number of bytes per pixel th
24 void Read(const char *filename, byte **pixels, int32 *width, int32 *height, int32 *bytesperpixel)
25 {
26     // To read the file in binary mode
27     FILE *imageFile = fopen(filename, "rb");
28     // To read data offset
29     int32 dataOffset;
30     fseek(imageFile, DATA_OFFSET_OFFSET, SEEK_SET);
31     fread(&dataOffset, 4, 1, imageFile);
32     // To read width
33     fseek(imageFile, WIDTH_OFFSET, SEEK_SET);
34     fread(width, 4, 1, imageFile);
35     // To read height
36     fseek(imageFile, HEIGHT_OFFSET, SEEK_SET);
```



```
37     // To read height
38     fseek(imageFile, HEIGHT_OFFSET, SEEK_SET);
39     fread(height, 4, 1, imageFile);
40     // To read bits per pixel
41     int16 bitsPerPixel;
42     fseek(imageFile, BITS_PER_PIXEL_OFFSET, SEEK_SET);
43     fread(&bitsPerPixel, 2, 1, imageFile);
44     // To allocate a pixel array
45     *bytesperpixel = ((int32)bitsPerPixel) / 8;
46
47     // Rows are stored bottom-up, Each row is padded to be a multiple of 4 bytes
48     // Calculated padded row size in bytes
49     int paddedRowSize = (int)(4 * ceil((float)(*width) / 4.0f)) * (*bytesperpixel);
50     // We are not interested in the padded bytes, so we allocate memory just for the pixel data
51     int unpaddedRowSize = (*width) * (*bytesperpixel);
52     // Size of pixel data in bytes
53     int totalSize = unpaddedRowSize * (*height);
54     *pixels = (byte *)malloc(totalSize);
55     // To read the pixel data Row by Row. Data is padded and stored bottom-up
56     // Pointing to the last row of unpadded pixel array
57     byte *currentRowPointer = *pixels + ((*height - 1) * unpaddedRowSize);
58     for (int i = 0; i < *height; i++)
59     {
60         // To put file cursor in the next row from top to bottom
61         fseek(imageFile, dataOffset + (i * paddedRowSize), SEEK_SET);
62         // To read only unpaddedRowSize bytes (we can ignore the padding bytes)
63         fread(currentRowPointer, 1, unpaddedRowSize, imageFile);
64         // Pointing to the next row (from bottom to top)
65         currentRowPointer -= unpaddedRowSize;
66     }
67     fclose(imageFile);
68 }
```

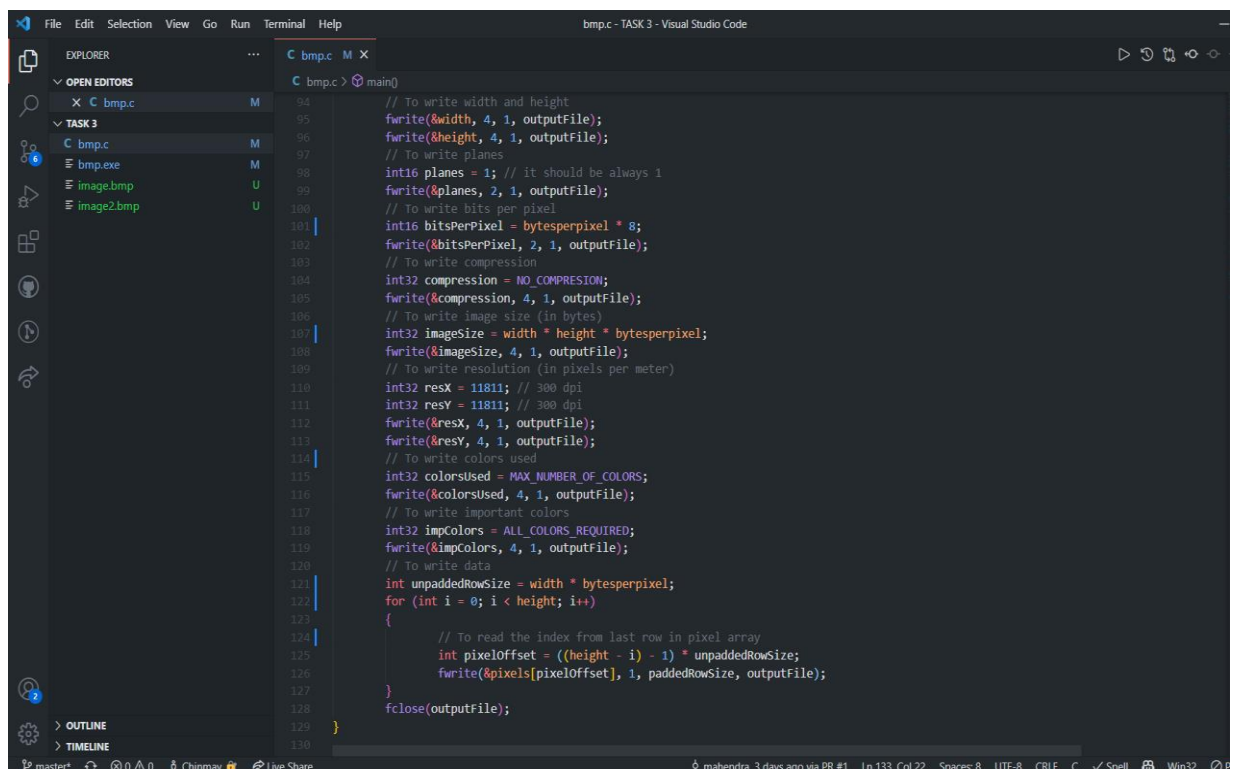
2. Function to Write Bitmap Image



```
File Edit Selection View Go Run Terminal Help
bmp.c - TASK 3 - Visual Studio Code

EXPLORER
OPEN EDITORS
  C bmp.c M X
TASK 3
  C bmp.c M
  bmp.exe M
  image.bmp U
  image2.bmp U

C bmp.c
68 // Inputs :
69 // filename: Name of the file to save, pixels: Pointer to the pixel data array, width: Width of the image in pixels
70 // height: Height of the image in pixels, bytesperpixel: Number of bytes per pixel that are used in the image
71 void Write(const char *filename, byte *pixels, int32 width, int32 height, int32 bytesperpixel)
72 {
73     FILE *outputFile = fopen(filename, "wb");
74     // HEADER // (Write signature)
75     const char *BM = "BM";
76     fwrite(&BM[0], 1, 1, outputFile);
77     fwrite(&BM[1], 1, 1, outputFile);
78     // To write file size considering padded bytes
79     int paddedRowSize = (int)(4 * ceil((float)width / 4.0f)) * bytesperpixel;
80     int32 fileSize = paddedRowSize * height + HEADER_SIZE + INFO_HEADER_SIZE;
81     fwrite(&fileSize, 4, 1, outputFile);
82     // To Write reserved
83     int32 reserved = 0x0000;
84     fwrite(&reserved, 4, 1, outputFile);
85     // To write data offset
86     int32 dataOffset = HEADER_SIZE + INFO_HEADER_SIZE;
87     fwrite(&dataOffset, 4, 1, outputFile);
88     // Info Header //
89     // To write size
90     int32 infoHeaderSize = INFO_HEADER_SIZE;
91     fwrite(&infoHeaderSize, 4, 1, outputFile);
92     // To write width and height
93     fwrite(&width, 4, 1, outputFile);
94     fwrite(&height, 4, 1, outputFile);
95     // To write planes
96     int16 planes = 1; // it should be always 1
97     fwrite(&planes, 2, 1, outputFile);
98     // To write bits per pixel
99     int16 bitsPerPixel = bytesperpixel * 8;
100     fwrite(&bitsPerPixel, 2, 1, outputFile);
101     // To write compression
102     int32 compression = NO_COMPRESSION;
103     fwrite(&compression, 4, 1, outputFile);
104     // To write image size (in bytes)
105     int32 imageSize = width * height * bytesperpixel;
106     fwrite(&imageSize, 4, 1, outputFile);
107     // To write resolution (in pixels per meter)
108     int32 resX = 11811; // 300 dpi
109     int32 resY = 11811; // 300 dpi
110     fwrite(&resX, 4, 1, outputFile);
111     fwrite(&resY, 4, 1, outputFile);
112     // To write colors used
113     int32 colorsUsed = MAX_NUMBER_OF_COLORS;
114     fwrite(&colorsUsed, 4, 1, outputFile);
115     // To write important colors
116     int32 impColors = ALL_COLORS_REQUIRED;
117     fwrite(&impColors, 4, 1, outputFile);
118     // To write data
119     int unpaddedRowSize = width * bytesperpixel;
120     for (int i = 0; i < height; i++)
121     {
122         // To read the index from last row in pixel array
123         int pixelOffset = ((height - i) - 1) * unpaddedRowSize;
124         fwrite(&pixels[pixelOffset], 1, paddedRowSize, outputFile);
125     }
126     fclose(outputFile);
127 }
```



```
File Edit Selection View Go Run Terminal Help
bmp.c - TASK 3 - Visual Studio Code

EXPLORER
OPEN EDITORS
  C bmp.c M X
TASK 3
  C bmp.c M
  bmp.exe M
  image.bmp U
  image2.bmp U

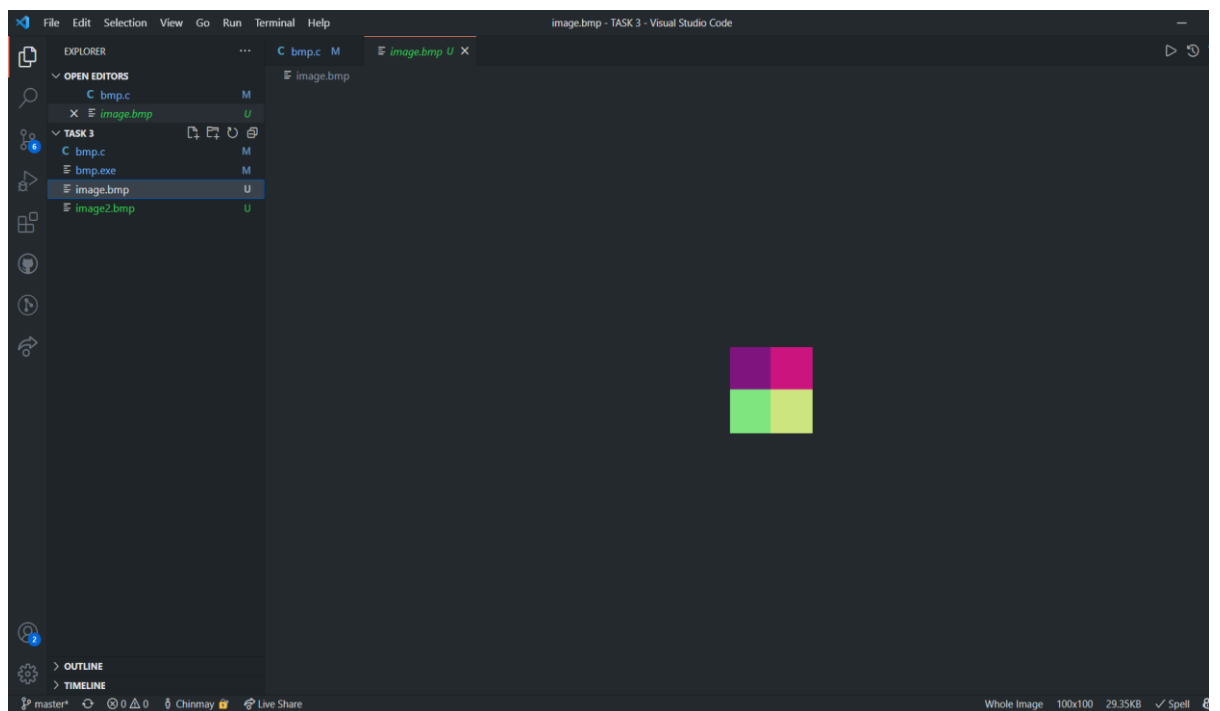
C bmp.c
94 // To write width and height
95 fwrite(&width, 4, 1, outputFile);
96 fwrite(&height, 4, 1, outputFile);
97 // To write planes
98 int16 planes = 1; // it should be always 1
99 fwrite(&planes, 2, 1, outputFile);
100 // To write bits per pixel
101 int16 bitsPerPixel = bytesperpixel * 8;
102 fwrite(&bitsPerPixel, 2, 1, outputFile);
103 // To write compression
104 int32 compression = NO_COMPRESSION;
105 fwrite(&compression, 4, 1, outputFile);
106 // To write image size (in bytes)
107 int32 imageSize = width * height * bytesperpixel;
108 fwrite(&imageSize, 4, 1, outputFile);
109 // To write resolution (in pixels per meter)
110 int32 resX = 11811; // 300 dpi
111 int32 resY = 11811; // 300 dpi
112 fwrite(&resX, 4, 1, outputFile);
113 fwrite(&resY, 4, 1, outputFile);
114 // To write colors used
115 int32 colorsUsed = MAX_NUMBER_OF_COLORS;
116 fwrite(&colorsUsed, 4, 1, outputFile);
117 // To write important colors
118 int32 impColors = ALL_COLORS_REQUIRED;
119 fwrite(&impColors, 4, 1, outputFile);
120 // To write data
121 int unpaddedRowSize = width * bytesperpixel;
122 for (int i = 0; i < height; i++)
123 {
124     // To read the index from last row in pixel array
125     int pixelOffset = ((height - i) - 1) * unpaddedRowSize;
126     fwrite(&pixels[pixelOffset], 1, paddedRowSize, outputFile);
127 }
128 fclose(outputFile);
129 }
130 }
```

3. Main Function

```
130
131  int main()
132  {
133      byte *pixels;
134      int32 width;
135      int32 height;
136      int32 bytesPerPixel;
137      Read("image.bmp", &pixels, &width, &height, &bytesPerPixel);
138      Write("image2.bmp", pixels, width, height, bytesPerPixel);
139      free(pixels);
140      return 0;
141  }
```

Visual Studio Code interface showing the main function code. The Explorer sidebar on the left shows the file structure with 'image.bmp' selected. The status bar at the bottom indicates the current file is 'image.bmp' at line 133, column 22.

4. Input (Read) Bitmap Image



5. Output (Write) Bitmap Image

