

|  |  |
| --- | --- |
| 成  绩 |  |



**课 程 名 称**  现代图像处理技术

**题 目 名 称** 位图(BMP)文件结构分析及

文件读取 、显示与保存

**学 院** 计算机学院

**专 业 班 级** 电子信息专硕1班

**学 生 姓 名** 梁增国

**学 号**  2112005119

**指 导 老 师**  战荫伟

**日 期** 2020年10月7日

**实验报告**

**目 录**

[1. 摘要 3](#_Toc52981101)

[2. 实验内容与相关平台 3](#_Toc52981102)

[2.1 实验内容 3](#_Toc52981103)

[2.2 实验的相关平台与工具 3](#_Toc52981104)

[3. BMP位图文件结构分析 3](#_Toc52981105)

[3.1 位图的文件结构 3](#_Toc52981106)

[3.2 位图的文件头分析 4](#_Toc52981107)

[3.3 位图的信息头分析 5](#_Toc52981108)

[3.4 调色板分析 7](#_Toc52981109)

[3.5 位图数据分析 9](#_Toc52981110)

[4. 使用C++实现对位图文件的读写、显示 10](#_Toc52981111)

[4.1 位图文件的读取 10](#_Toc52981112)

[4.2 在控制台上显示位图图像 11](#_Toc52981113)

[4.3 位图文件的保存 12](#_Toc52981114)

[4.4 实验效果 13](#_Toc52981115)

[5. 总结 13](#_Toc52981116)

[6. 参考文章 14](#_Toc52981117)

# 摘要

BMP是英文Bitmap（位图）的简写，它是Windows操作系统中的标准图像文件格式，能够被多种Windows应用程序所支持。BMP图像文件是Windows采用的图形文件格式，在Windows环境下运行的所有图象处理软件都支持BMP图象文件格式。Windows系统内部各图像绘制操作都是以BMP为基础的。

本实验将会对BMP位图的文件结构进行分析，并实现BMP位图的读取、显示和保存。

**关键词：BMP位图文件结构分析、BMP位图的读取、BMP位图的保存、BMP位图的显示**

# 实验内容与相关平台

## 实验内容

* 对BMP位图的文件结构进行分析
* 编写程序，实现对24位彩色位图进行读取、显示、保存操作

## 实验的相关平台与工具

Notepad++（用于分析BMP图像的文件结构）、Vs code、C++

# BMP位图文件结构分析

在本节中，我们将要对BMP位图的文件结构进行分析。在此之前，我们需要事先了解两个关键点：

* 在BMP文件中，数据存储采用**小端方式(little endian)**，即“低地址存放低位数据，高地址存放高位数据”。
* 以下所有分析均**以字节为单位**进行。

## 位图的文件结构

位图的文件结构如表3-1所示，位图的数据包括四项，分别是：**位图文件头、位图信息头、调色板和位图数据**。

|  |
| --- |
| 位图文件头BITMAPFILEHEADER |
| 位图信息头BITMAPINFOHEADER |
| 调色板Palette |
| 位图数据ImageData |

表3-1

## 位图的文件头分析

位图文件头主要用于识别位图文件，以及记录文件的大小、位图数据位置等信息，共占14个字节。图3-2是位图文件头结构的定义：

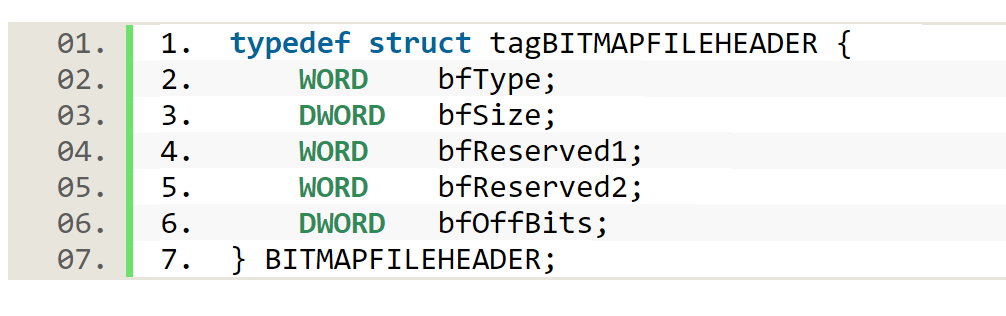


图3-2

位图文件头的字段含义如表3-3所示：

|  |  |  |
| --- | --- | --- |
| 字段 | 字节数 | 含义 |
| bfType | 2 | 声明位图文件的类型，该值必须为0x4D42，即字符'BM'。表示这是Windows支持的位图格式。  【注】该值也可以设置位’BA’,’CI’,’CP’等不同格式，但由于因为OS/2系统并没有被普及开。因此在编程时，只需判断第一个标识为否为“BM”即可。 |
| bfSize | 4 | 声明BMP文件的大小，单位是字节 |
| bfReserved1 | 2 | 保留字段，必须设置为0 |
| bfReserved2 | 2 | 保留字段，必须设置为0 |
| bfOffBits | 4 | 声明从文件头开始到实际的图象数据之间的字节的偏移量，可以用这个偏移值迅速的从文件中读取到位数据。 |

表3-3

用Notepad++打开BMP图像文件“lena-单色位.bmp”，如下图3-4所示。可见红框1中，第1-2字节数据为0x4d42，为BMP位图的固定标识。在红框2中，第3-6字节数据为0x00008d8e，表示36238字节，可见该值与在Window资源管理器中查看文件属性中的图片大小的是一致的。

在红框3中，此处的数据为0x0000003e，表示62字节，表示位图数据位于从文件开始往后数的62字节处。

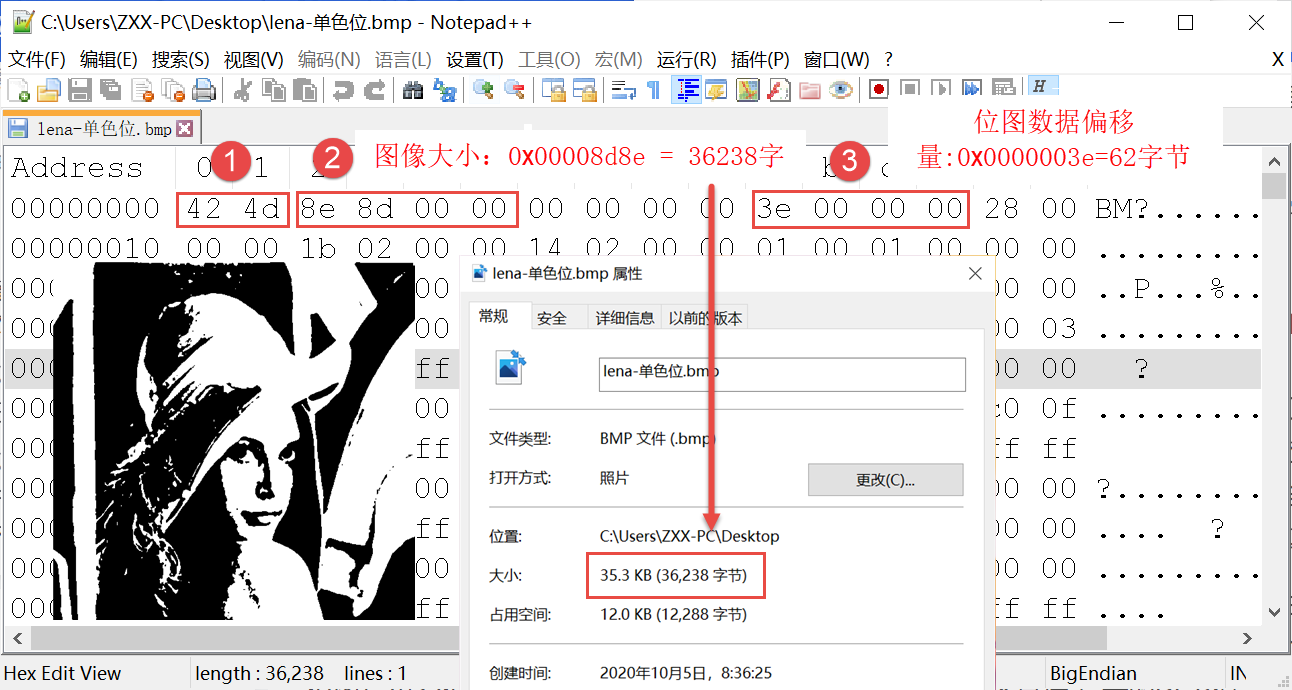


图3-4

## 位图的信息头分析

BITMAPINFO段由两部分组成：BITMAPINFOHEADER结构体和RGBQUAD结构体，其中的BITMAPINFOHEADER结构体表示位图信息头。同样地，Windows为位图信息头定义了如下结构体，如下图3-5所示：

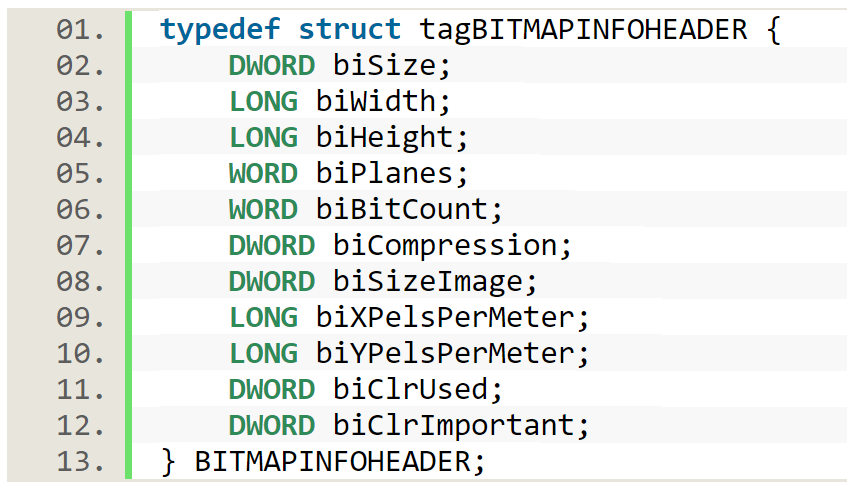


图3-5

位图信息头的字段含义如下表3-6所示：

|  |  |  |
| --- | --- | --- |
| 字段 | 占字节数 | 含义 |
| biSize | 4 | 声明BITMAPINFOHEADER占用的字节数 |
| biWidth | 4 | 声明图片的宽度，单位是像素 |
| biHeight | 4 | 声明图片的高度，单位是像素 |
| biPlanes | 2 | 声明目标设备说明位面数，其值将总是被设为1 |
| biBitCount | 2 | 声明单位像素的位数，表示Bmp图像的颜色位数，如24位图，32位图 |
| biCompression | 4 | 声明图像压缩属性，由于bmp图片是不压缩，该值等于0 |
| biSizeImage | 4 | 声明Bmp图像数据区的大小 |
| biXPelsPerMeter | 4 | 声明图像的水平分辨率 |
| biYPelsPerMeter | 4 | 声明图像的垂直分辨率 |
| biClrUsed | 4 | 声明使用了颜色索引表的数量 |
| biClrImportant | 4 | 声明重要的颜色的数量，等于0时表示所有颜色都很重要 |

表3-6

继续用Notepad++分析BMP图像的文件结构，如下图3-7所示。可见红框1所示的数据为biSize字段，它的值为0x00000028=40，表示位图信息头的大小为40字节。红框2与3所示的数据表示图像的宽度与高度，对应的值为0x0000 021b = 539像素，0x0000 0214 = 532像素。红框4处表示图像的位深度，因为这是一张黑白图像，所以位深度为1。

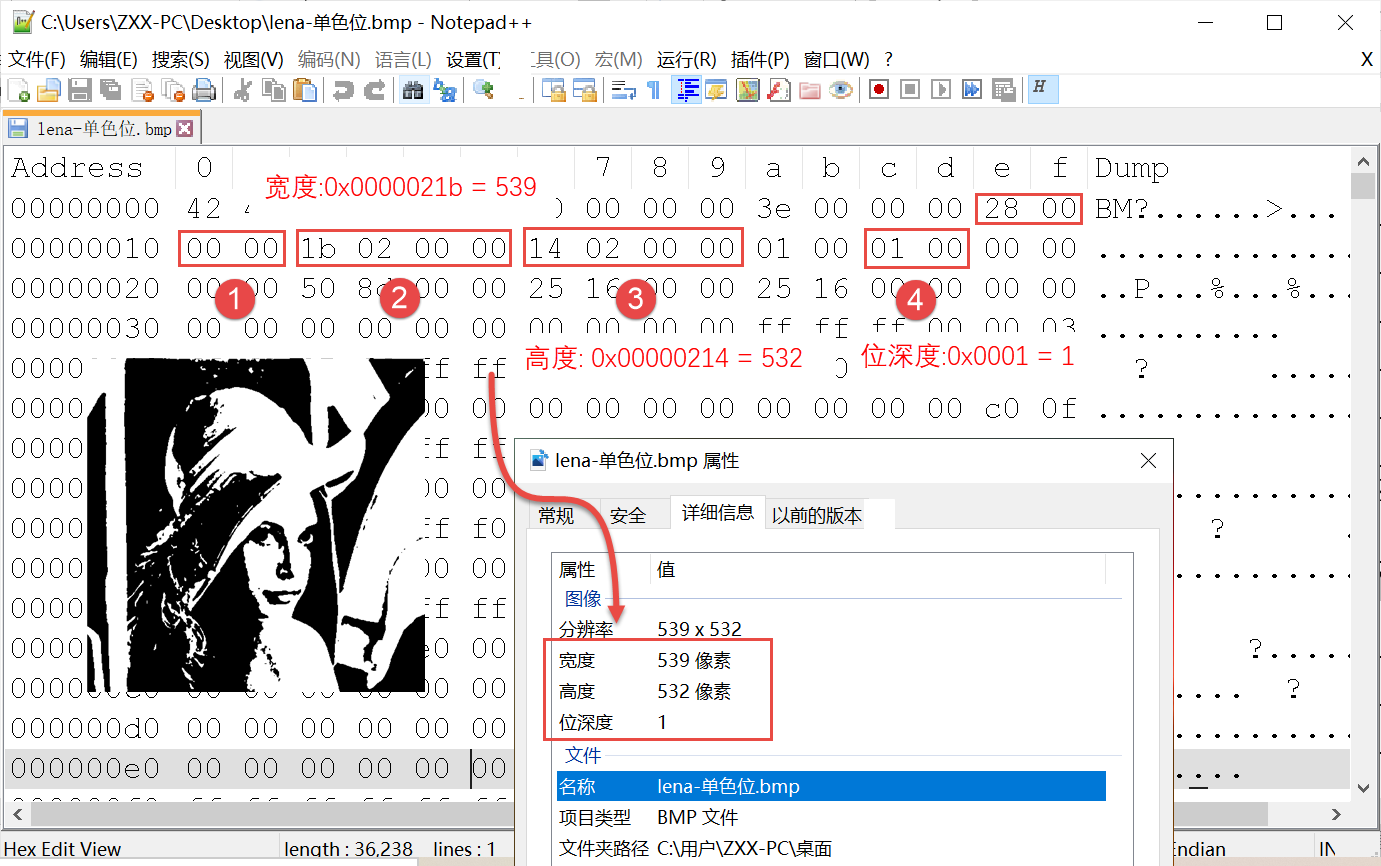


图3-7

若打开的是24位深度的图片，可见该字段的值为0x0018，代表颜色深度为24，如下图3-8所示。

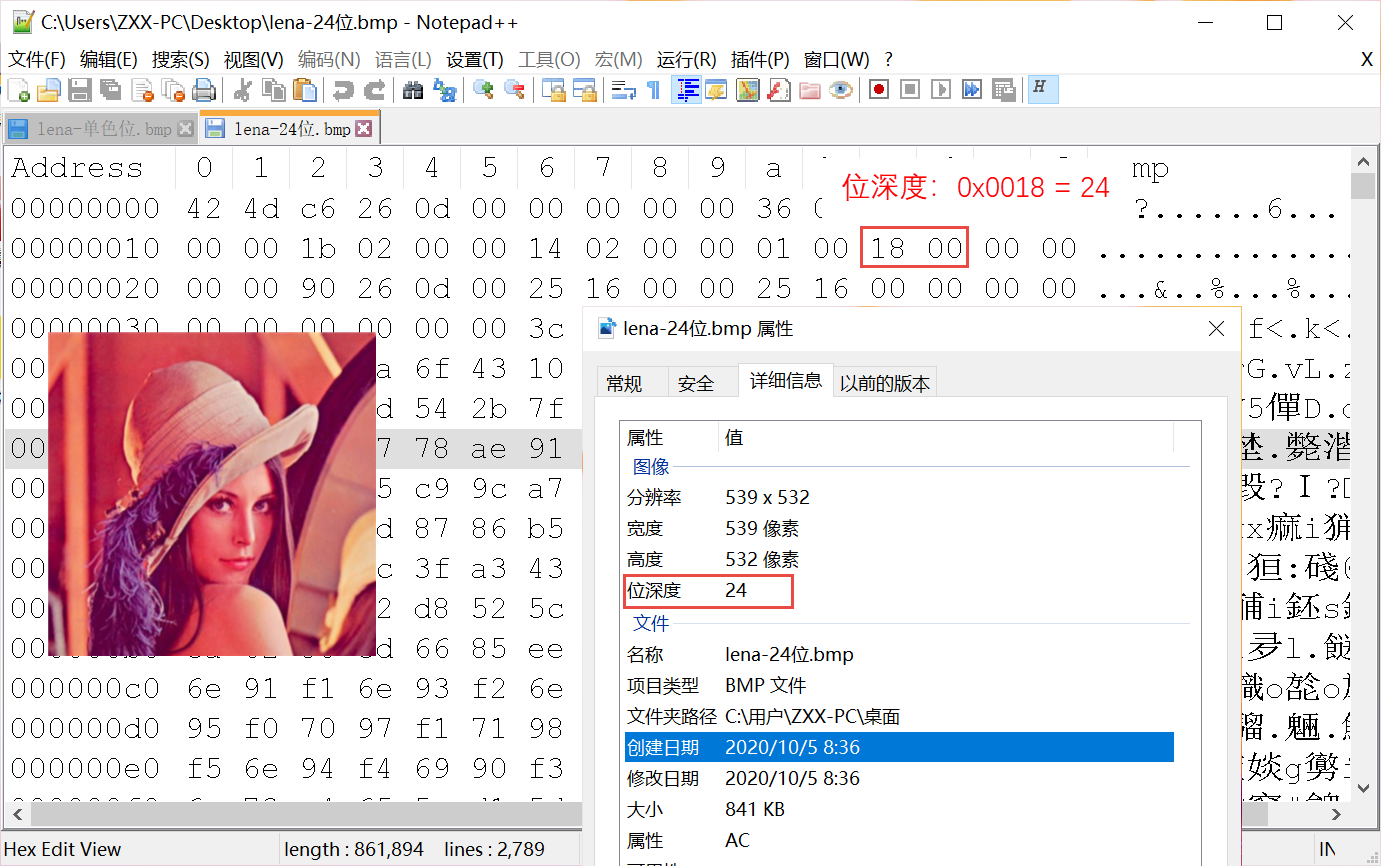


图3-8

继续分析文件，如下图3-9所示。红框5处声明了BMP图像的数据区大小，即0x00008d50 = 36176字节。红框6处定义了图像的水平分辨率和垂直分辨率，红框7处定义了使用彩色表的索引值的数量，当该值为0时，表示使用所有调色板项。

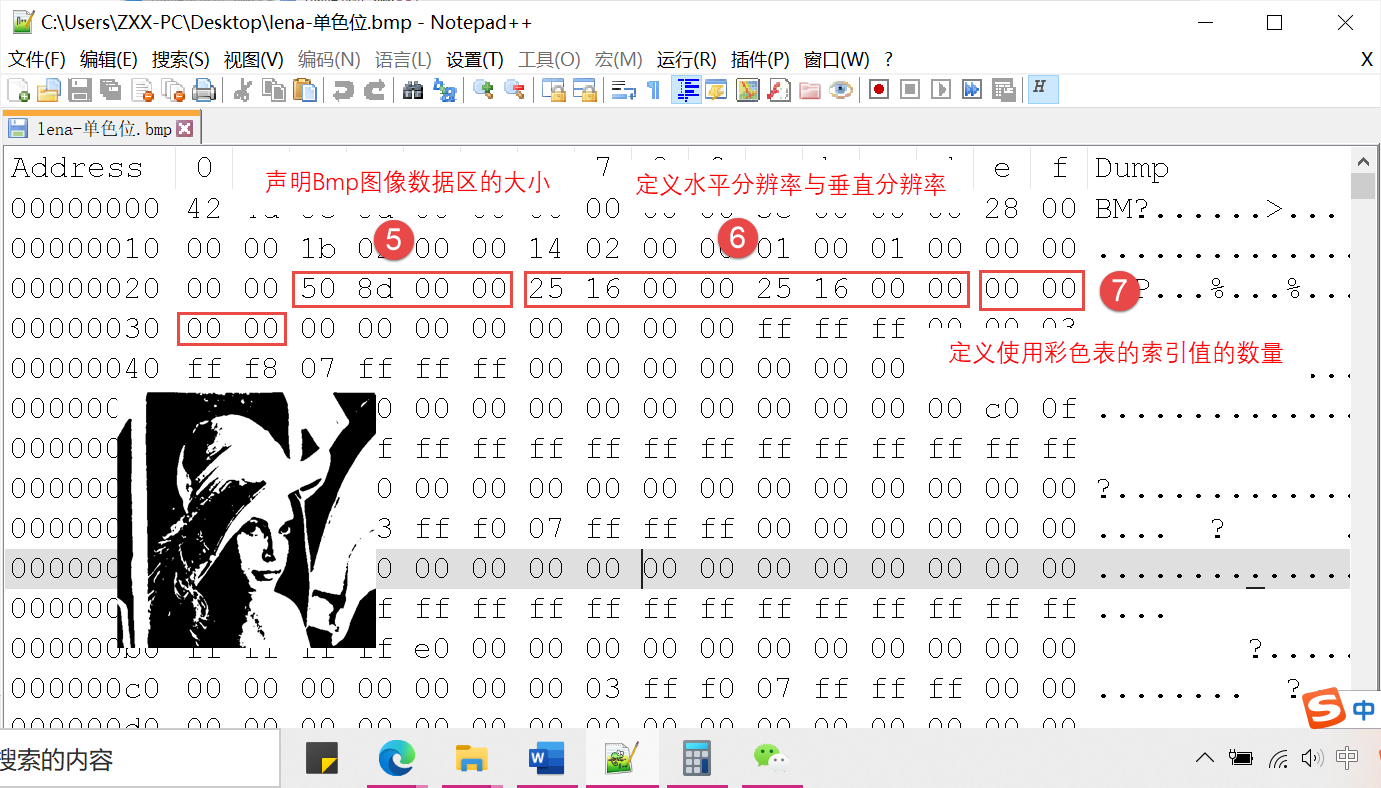


图3-9

## 调色板分析

调色板一般是针对16位以下的图像设置的，对于16位及以上的BMP格式图像，其位图像素数据是直接对应像素的RGB颜色值进行描述，因此省去了调色板。对于16位以下的BMP格式图像，其位图像素数据中记录的是调色板的索引值。调色板的作用是，当图像的位深度值比较小时，通过调色板记录所有的颜色值，而位图数据则存储调色板的索引项，因此达到节省存储空间的效果。

调色板的数据由RGBQUAD结构体项组成，该结构体由4个字节型数据组成，所以一个RGBQUAD结构体只占用4字节空间，从左到右每个字节依次表示(蓝色，绿色，红色，未使用)。调色板的结构体定义如下图3-10所示：

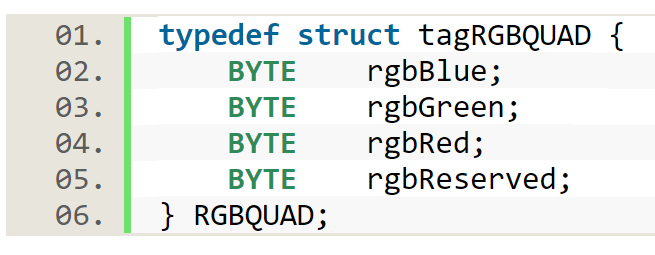


图3-10

分析图像的第55-62个字节，该处声明的是图像的彩色表项，由于现在使用的图像是单色图，只有黑白两种颜色，所以调色板中也只有两项，对应着黑色和白色。如下图3-11所示。

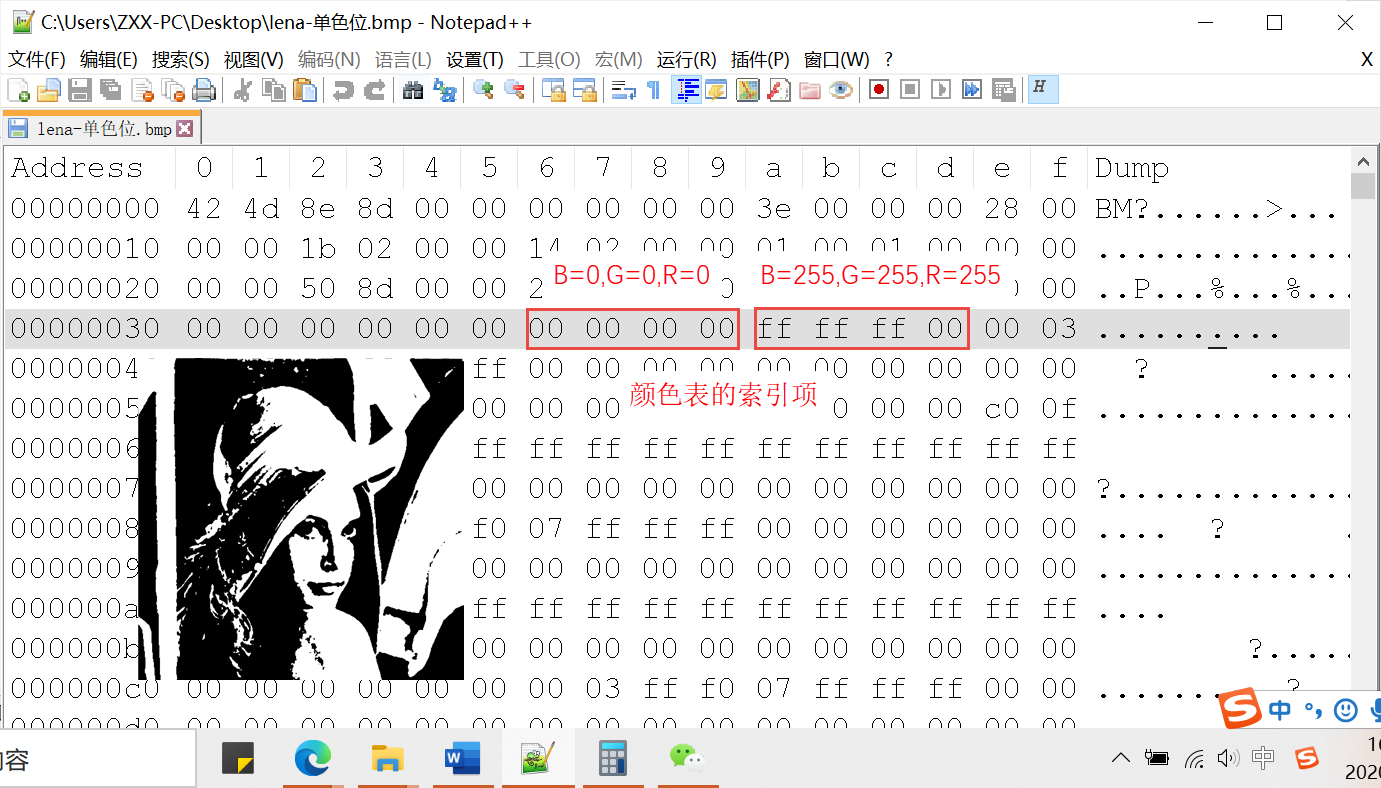


图3-11

接下来，我们看一下位深度大于16位的BMP图像的调色板。

我们用Notepad++打开一张位深度为24的BMP图像，如下图3-12所示。红框1处为位图文件头的bfOffBits字段，值为54字节，表示从文件头起始到位图数据之间的字节的偏移量54字节。

红框2处的字段为位图信息头的biSize字段，值为40字节。观察两组数据数据，位图的文件头固定为14字节，加上信息头的40字节因此总字节数为54字节，正等于bfOffBits字段的偏移量。可以由此得知，24位位深度的BMP图像没有调色板数据。

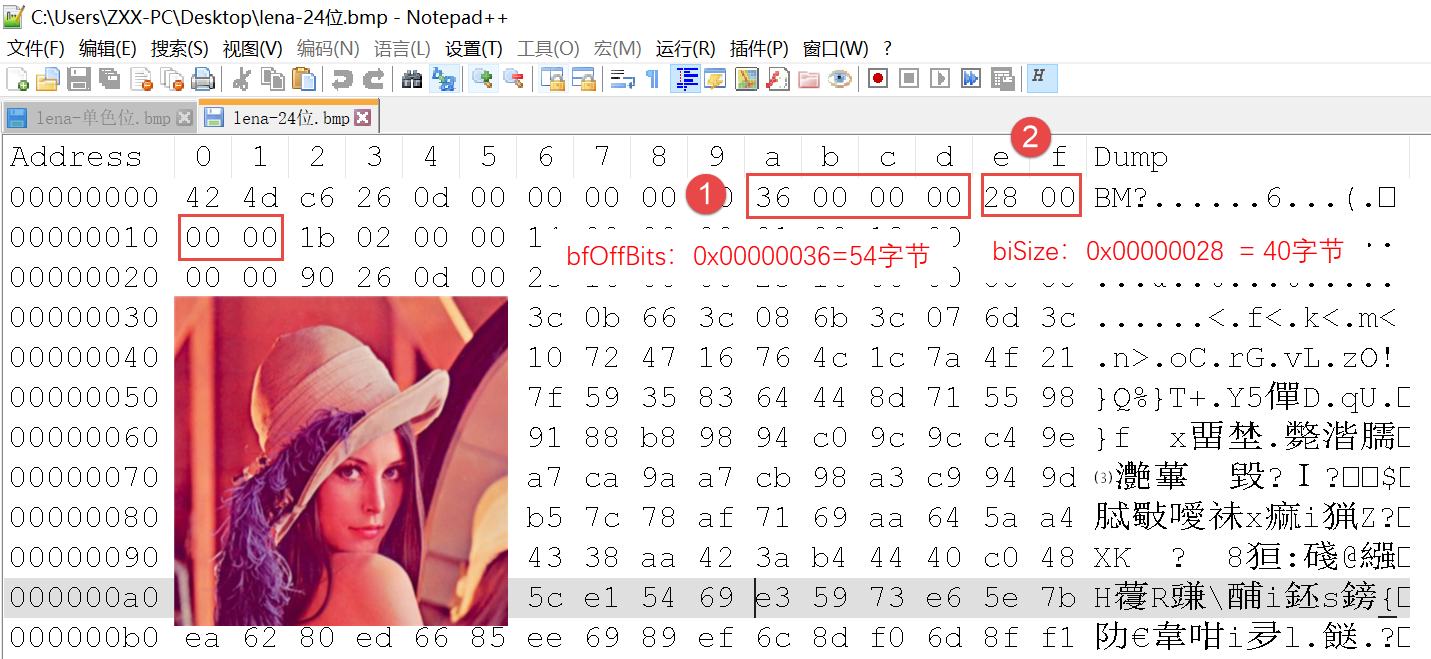


图3-12

## 位图数据分析

位图数据记录了位图每一个像素的像素值，**存储的顺序是在扫描行内是从左到右，扫描行之间是从下到上**。根据不同的位图，位图数据所占据的字节数也是不同的。比如，对于24位的位图，每三个字节表示一个像素。对于本案例中的单色图，一个字节则可以对应八个像素点的像素值。

根据图像提供的位图数据，可以得知每个像素点的值，以此绘制图像。

如下图3-12所示，位图数据共有36176字节，位图文件头与位图信息头共54字节，再加上彩色表的两个索引项共8个字节，可以得知该图像共36238字节。此数据与用Window资源管理器直接查看图像的大小一致。

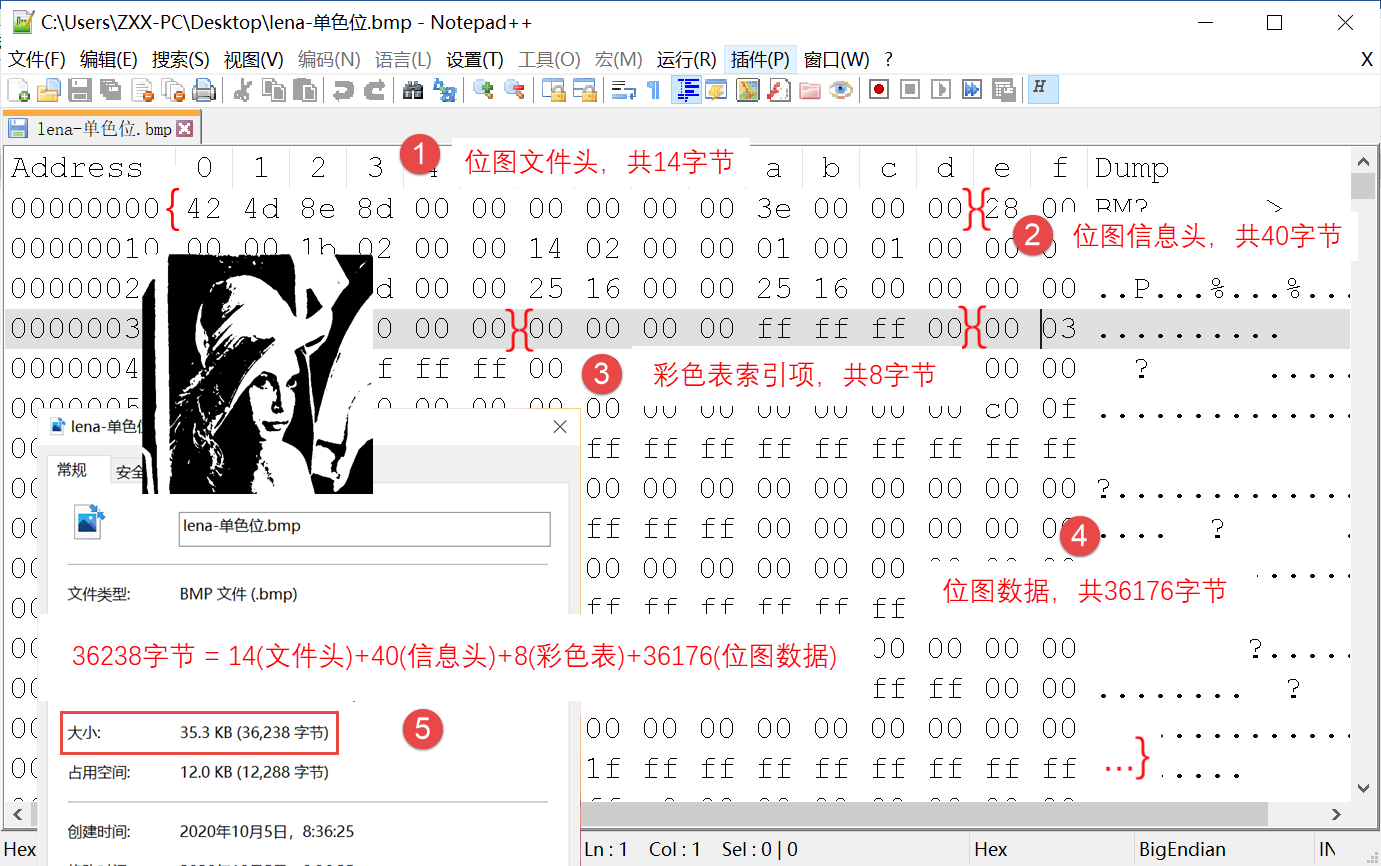


图3-12

在位图数据存储与读取过程中，有一点需要特别注意：**BMP存储格式要求每行的字节数必须是4的倍数。**若某行的字节数不是4的倍数，需要额外添加字符‘0’凑够到4的倍数。在对位图数据进行读写时，这一点需要特别留意，否则无法对位图图像进行正确的读写。

# 使用C++实现对位图文件的读写、显示

在本小节中，将会用C++语言实现对24位位图文件的读写，显示操作。

本程序定义了一个新的结构体ImgInfo，里面包含了位图文件头BITMAPFILEHEADER、位图信息头BITMAPINFOHEADER，还有一个二维数组imgData，用于存放像素值信息。如下图4-1所示，加入二维数组imgData字段的好处是可以使用二维数组更方便地对图像的像素点进行操作。

在程序的主函数中，调用了readBitmap、showBitmap、saveBitmap三个函数，实现对BMP图像的读取、显示、保存操作，如图4-2所示。

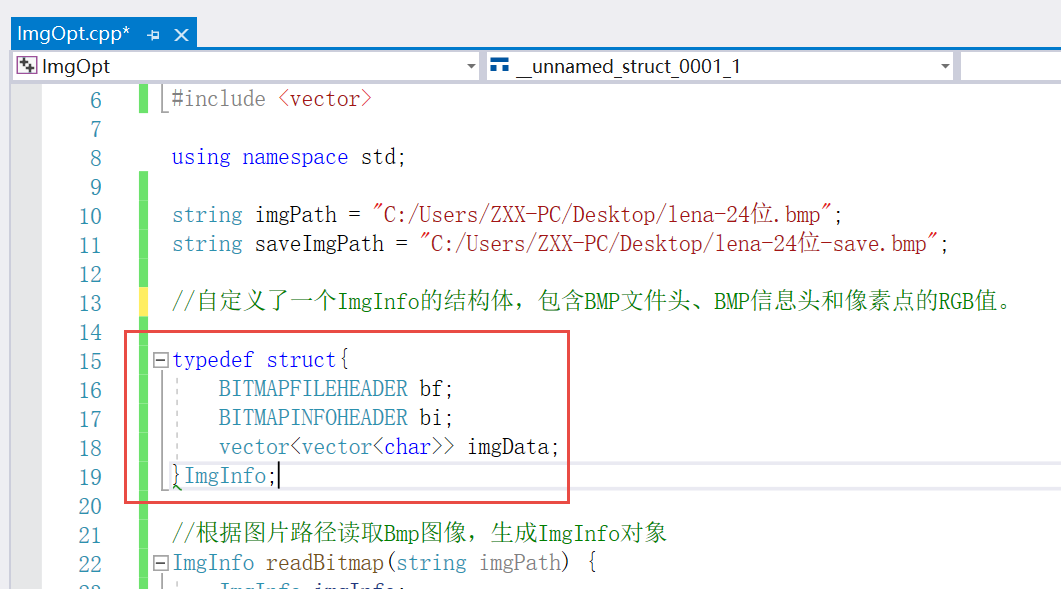


图4-1

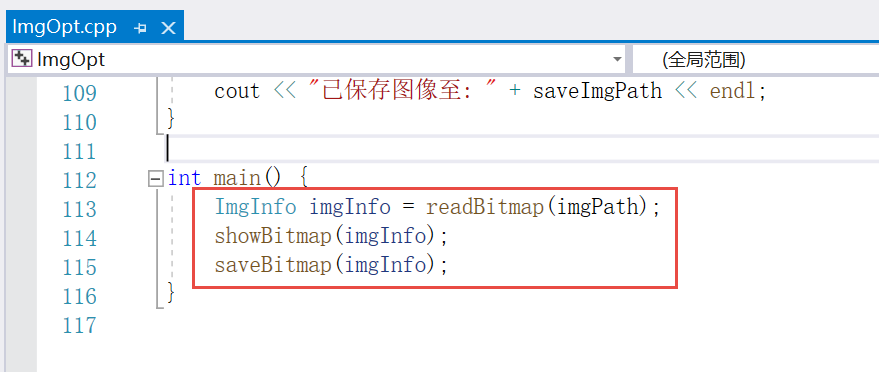


图4-2

## 位图文件的读取

位图文件读取主要由程序中的readBitmap函数实现，关键代码如下图4-3所示，通过使用fread函数实现对位图文件头与位图信息头的读取。

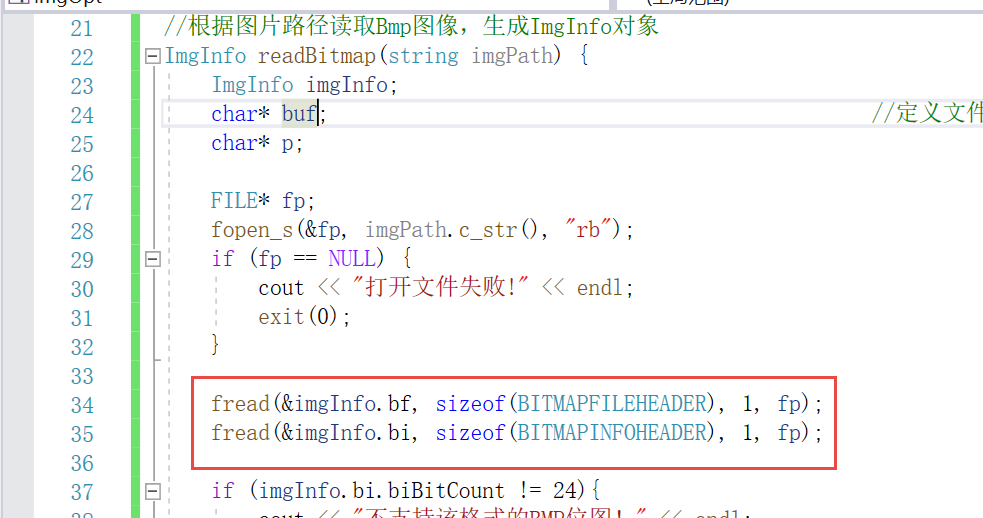


图4-3

在下图4-4中，通过fseek函数与位图文件头的bfOffBits字段，对图像像素数据进行定位，以此来读取像素数据信息，并存放到二维数组中。

注意蓝色框中的代码，由于BMP位图采用4字节对齐的存储机制，可能会存在一些无意义的填充数据，因此我们在读取数据时必须将他们排除。

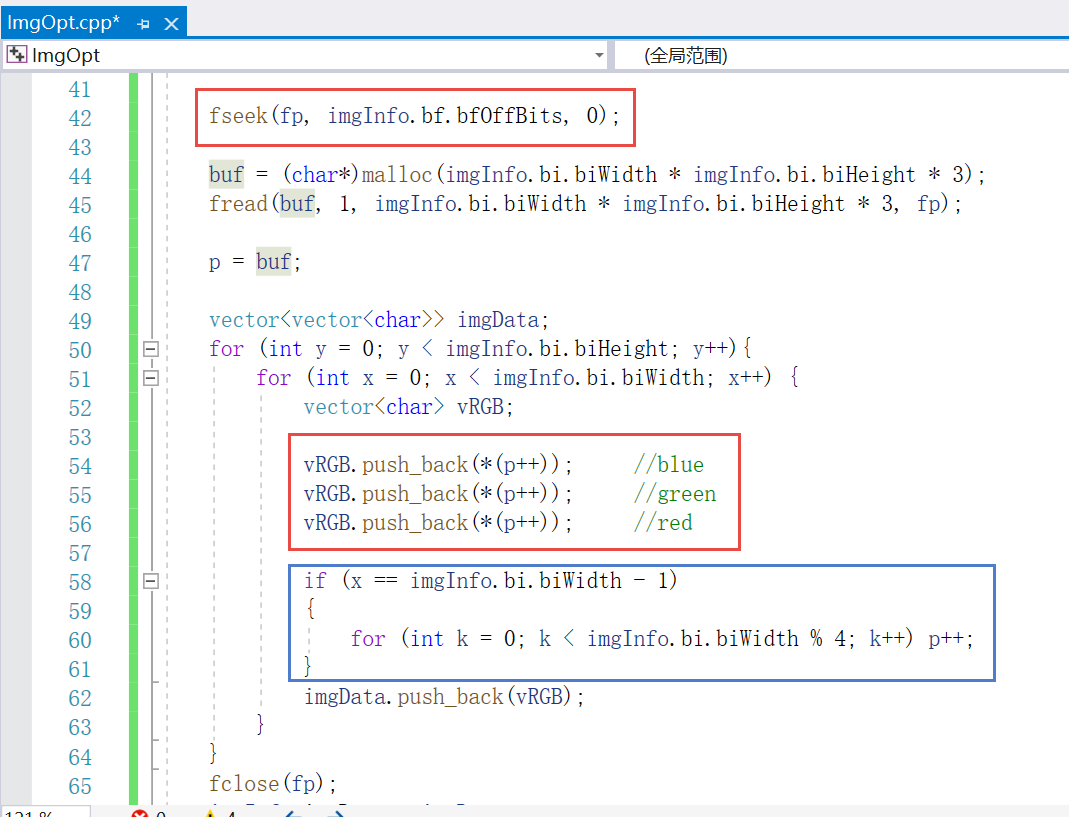


图4-4

## 在控制台上显示位图图像

在控制台上显示位图图像，主要由程序中的showBitmap函数实现。

根据结构体ImgInfo中的imgData字段，我们可以很轻易地获取图像的像素值信息，并使用SetPixel函数将像素值显示在控制台特定的位置，这部分的关键代码如下图4-5所示。

需要注意的是，**BMP位图的像素数据存储方式是行内从左到右，行间从下到上（即第一个数据存放的是图像左下角的像素信息，最后一个数据存放的是图像右上角的像素信息），**因此在编程时需要考虑清楚像素点与其图像实际的坐标位置。

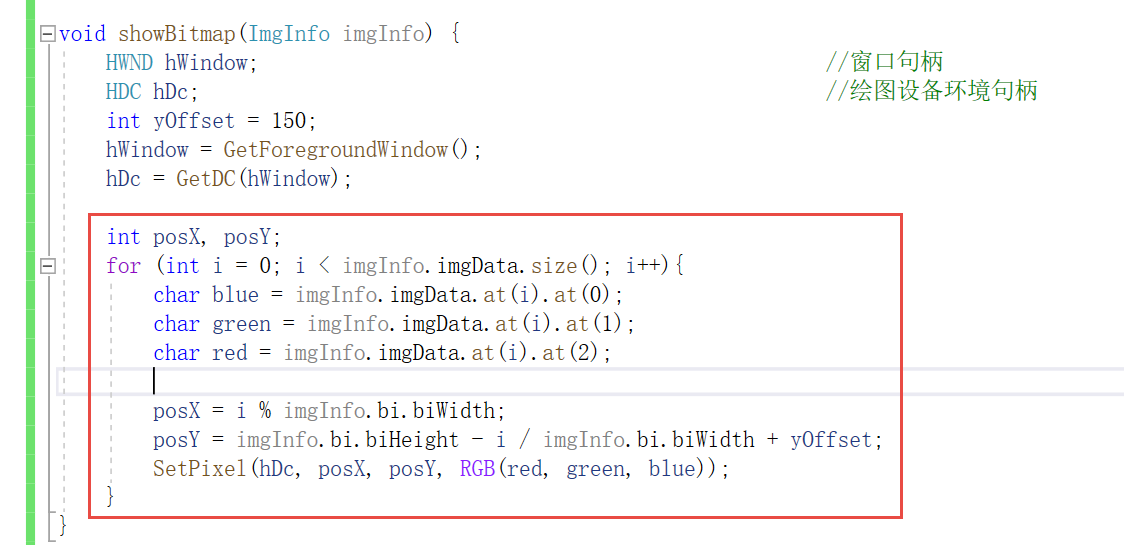


图4-5

## 位图文件的保存

位图文件的保存，主要在程序中的saveBitmap函数中实现，如下图4-6所示。与位图文件的读取类似，按照BMP位图的文件结构，先使用fwrite函数实现对位图文件头和位图信息头的写入，再遍历像素点信息将像素值写入文件中。

同样地，**位图的像素信息存取采用4字节对齐的方式**，在写入每一行位图数据后且字节长度不足4的倍数时，需要填充’0’字符。

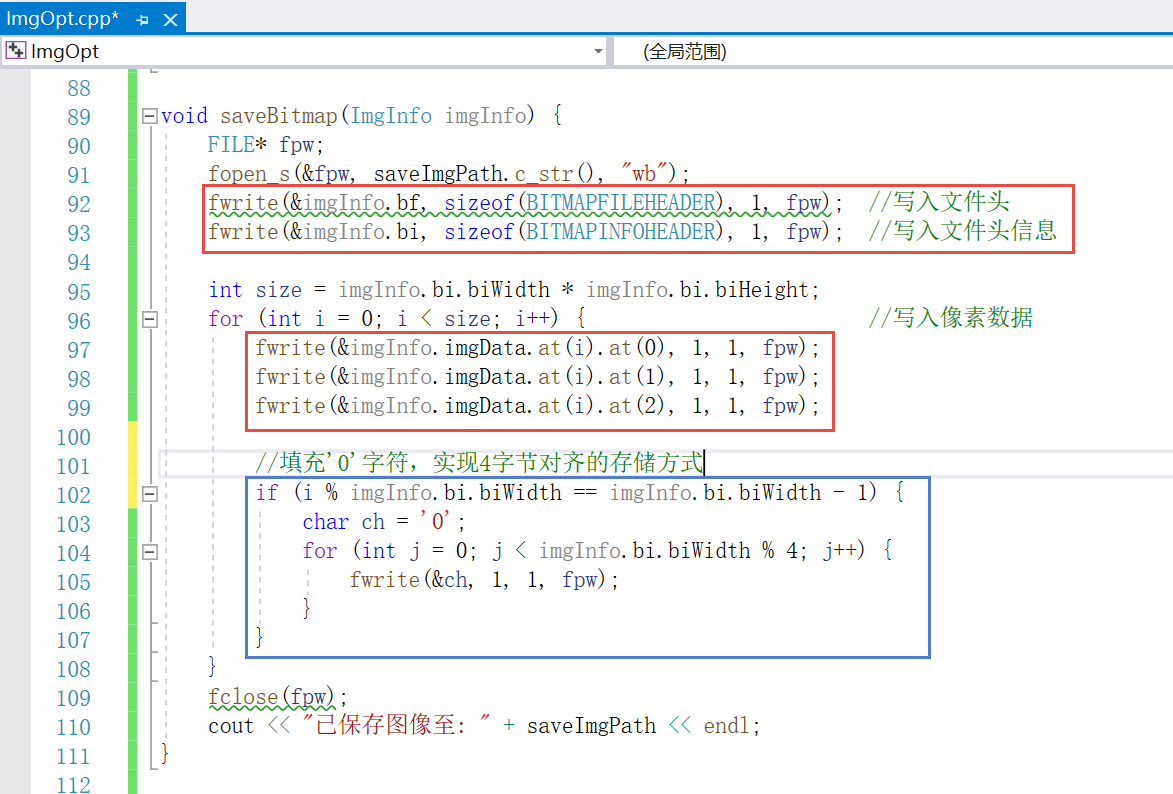


图4-6

## 实验效果

如下图4-7所示，在运行程序后，将图像数据读出，然后在控制台上显示图像，最后将图像保存到本地。

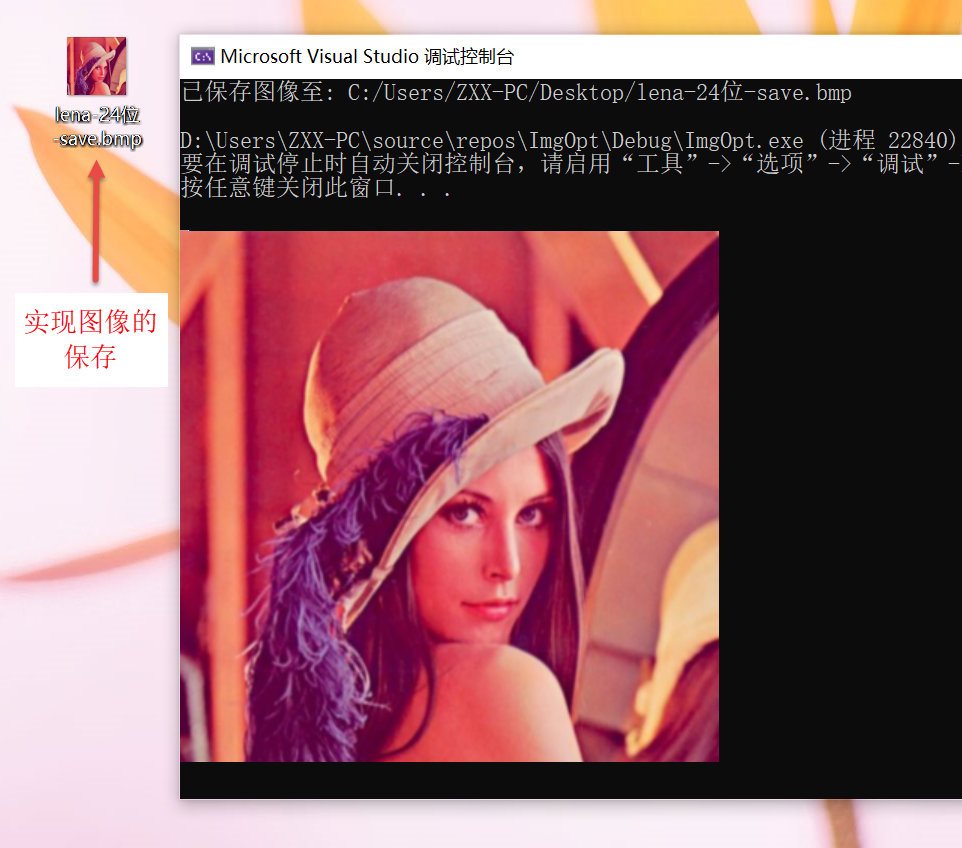


图4-7

# 总结

* 位图的文件结构包括四项，分别是：位图文件头、位图信息头、调色板和位图数据。
* 位图文件头存放位图文件的大小、位图数据位置等信息。
* 位图信息图存放位图文件的宽高、图像位深度、水平/垂直分辨率、位图数据大小等等关键信息。
* 调色板一般是针对16位以下的图像设置的，对于16位及以上的BMP格式图像，其位图像素数据是直接对应像素的RGB颜色值进行描述。
* 位图数据记录了位图每一个像素的像素值，存储的顺序是在扫描行内是从左到右，扫描行之间是从下到上，并要求每行的字节数必须是4的倍数。

# 参考文章

《百度百科--Bitmap位图》：

<https://baike.baidu.com/item/Bitmap/6493270?fr=aladdin>

《Bitmap 图片格式并用 C++ 读写 Bitmap》：

<https://blog.csdn.net/weixin_34208185/article/details/86257499>

《BMP格式详解》：

<https://blog.csdn.net/gwwgle/article/details/4775396>

《Bitmap每行4字节对齐》：

<https://blog.csdn.net/a_flying_bird/article/details/50585146>