**实习3 动态规划算法及应用**

**【题目一】BMP灰度图像压缩**

**一、实习目的与要求**

**【问题描述】**

灰度图像的像素值范围在[0，255]之间，如果采用一个像素一个字节的存储方式，势必会造成空间的浪费。如果采用一定的无损压缩算法，可以大大提高减小文件大小，减少存储空间。本课题要求针对提供的256色（8位）位图数据，采用教材上第15章动态规划中图像压缩算法（图像分段合并的思想），设计一个类，实现灰度位图数据的压缩和解压过程。

**【基本要求】**

一个完整的灰度图像类应具有以下功能：

（1）对8位位图数据的读功能，提供ReadBitmap方法。

ReadBitmap方法有一个参数为输入位图文件名(\*.bmp)，它能解析8位位图文件格式，获取位图BITMAPINFOHEADER信息和每个像素的数据信息，放入内存中。

（2）对8位位图数据的写功能，提供WriteBitmap方法。

WriteBitmap方法有一个参数为输出位图文件名(\*.bmp)，它能将内存中的位图文件信息，按照位图格式，写到位图文件中保存。

（3）灰度图像压缩功能，提供Compress方法。

Compress方法有一个参数为输出压缩文件名(\*.img) ，它能将已经装入到内存中的8位位图信息，进行压缩，形成段标题和以变长格式存储的像素的二进制串，写入到文件中（注意：Img文件格式自行定义）。

（4）灰度图像解压功能，提供UnCompress方法。

UnCompress方法有一个参数为输入压缩文件名(\*.img)，它能解析Img文件格式，将其在内存中解压缩为8位位图信息，以便输出为位图文件。

（5）以上是该灰度图像类基本的四个方法，在实现时可根据需要扩充其他方法。在设计时，要使用面向对象的思想，考虑各个成员的访问权限。

**【提高要求】**

（1）基于Windows对话框界面，可选择输入/输出文件名，有压缩进度条显示。

（2）采用不同的数据集，比较其压缩比，采用最有效的压缩方式。

**【测试数据】**

lena.bmp，512\*512\*8

**【测试用例】**

类的测试用例如下：

CCompressImage Test;

Test. ReadBitmap(“lena.bmp”); 读原始位图

Test. Compress(“Out.img”); 压缩

Test. UnCompress(“Out.img”); 解压

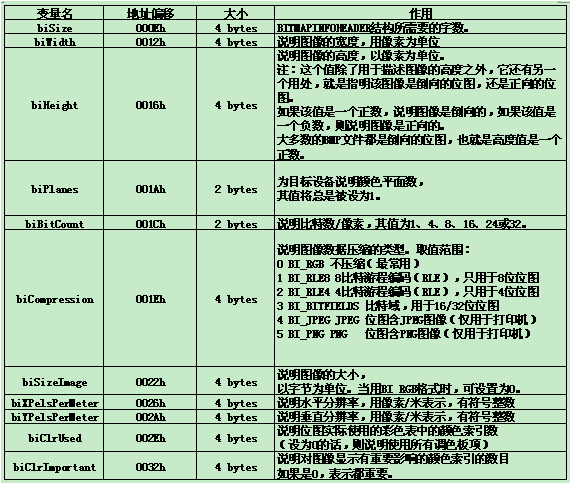
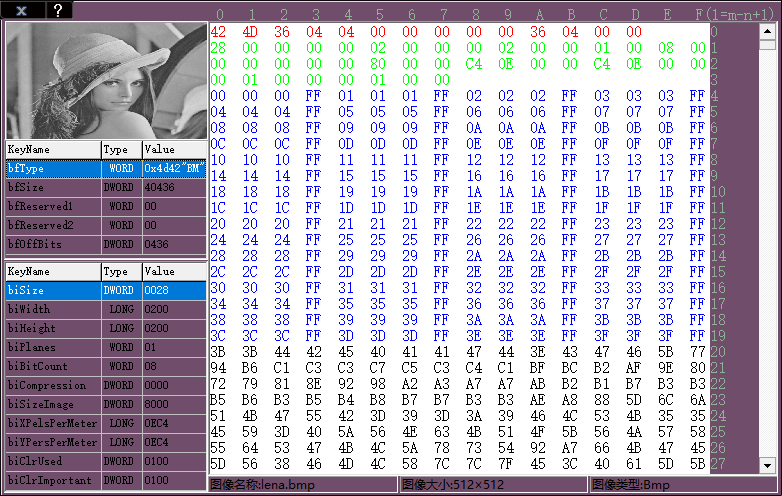
Test. WriteBitmap(“Out.bmp”); 还原位图信息

【测试结果】

可以使用MD5比较解压后的图与原图是否一样，验证你所实现的灰度图像类是否做到了无损压缩。

**【实现提示】**

有关8位的位图格式可以参考MSDN中BITMAPINFOHEADER结构的说明文档，注意其中biBitCount=8的说明。



**二、分析与设计**

1. **需求分析与类设计**

**本次实习采用QT来完成相应的操作**

**1.需要我们先对图读取bmp文件然后经过dp算法之后输出一个新的文件，我们将其后缀命令为mdp**

**2.需要我们读取mdp文件之后解压后输出可见文件。**

**类设计：**

**使用了**

**一.C\_Ucompress.h**

**①其中包含compress函数。**

**它的具体过程如下：**

1. **读取源图像文件，解析其中的BMP文件头、信息头和颜色表，以及像素数据，并将像素数据存储在二维unsigned char类型的数组Pix中。同时打开一个新的二进制文件，用于存储压缩后的图像数据。**
2. **将图像像素数据按“蛇形”方式重新排列，即第一行从左到右，第二行从右到左，以此类推，最后将所有行合并成一个一维unsigned int类型的数组d。**
3. **对d数组进行压缩处理，生成三个一维unsigned int类型的数组：cpsPic、bnyPic、ucpsPic。其中，cpsPic存储压缩后的像素数据，bnyPic存储每段数据的二进制位数，ucpsPic存储每段数据的长度。**
4. **对压缩后的数据进行分段，每段包含连续相同像素的像素值和其出现的次数。将每一段的长度、位数和像素数据分别写入输出文件中。**
5. **关闭输入和输出文件，返回true表示压缩成功。**

**②包含uncompress函数**

1. **它的具体过程如下：检查传入的文件名是否以".mdp"为后缀，如果不是，则直接返回false，表示无法解压缩该文件。**
2. **打开要解压缩的文件，读取其中的BITMAPFILEHEADER、BITMAPINFOHEADER和调色板信息，这些信息用于后面生成位图文件。**
3. **读取压缩文件的正文内容。具体的读取方式是将连续的若干位二进制数据解析成对应的数字，再按照规定的格式将数字转换成像素点的颜色值。由于压缩时采用了变长编码方式，因此解压缩时需要逐位解析。具体的解析方式可以参考代码中的注释。**
4. **将还原后的像素点颜色值写入新的位图文件中，并保存。如果保存成功，则返回true，表示解压缩成功；否则返回false，表示解压缩失败。**

**二：使用了dptocompress**

* 1. **其中包含opt函数：  
     他是找到问题的回溯解，其中没有使用递归，为了避免可能大量数据导致的一个栈溢出问题。**
  2. **其中还包含dp函数：**

**这段代码具体实现了BMP位图压缩算法中的“动态规划算法”部分。**

**该函数接受四个参数，分别是：**

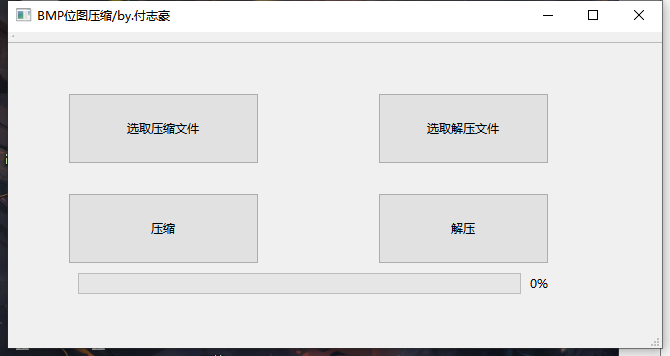
* **n：表示位图中像素点的总数；**
* **onlPic[]：表示原始位图中每个像素点的值；**
* **cpsPic[]：表示压缩后的位图中每个像素点所占用的比特数；**
* **ucpsPic[]：表示在当前像素点处采用“复制前面像素点”的压缩方式时，所复制的前面像素点的个数。**

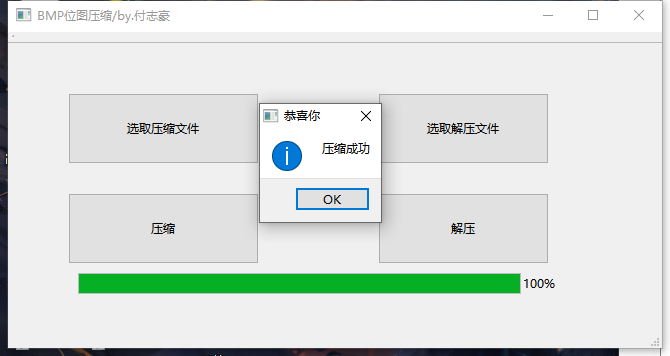
1. **算法设计与分析**

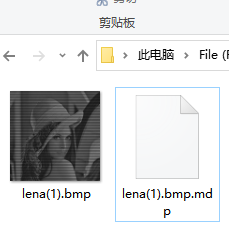
* **在这次实习之中主要使用了动态规划算法去做：  
  在dptocompress中的dp函数具体过程如下：**
* **首先，定义一个变量Lmax=256表示最大可以复制前面256个像素点。header=11表示BMP位图文件头占用11个字节。**
* **然后，对于第一个像素点，将它的cpsPic值设为0，表示不需要压缩；将它的ucpsPic值设为1，表示不需要复制前面的像素点。**
* **接着，对于第2到第n-1个像素点，采用动态规划的思想进行处理，具体来说，对于第i个像素点，它的压缩方式可能有两种：要么采用“复制前面像素点”方式，要么采用“存储二进制像素段”方式。因此，需要枚举前面256个像素点中的每一个点，计算在当前像素点处采用“复制前面像素点”的方式时所需要的比特数，以及在当前像素点处采用“存储二进制像素段”的方式时所需要的比特数，并比较两种方式的结果，选择比较优的一种方式。**
* **最后，得到每个像素点的压缩方式后，需要采用非递归方式对结果进行追溯，得到最终的压缩方案。**
* **总的来说，该函数的作用就是实现BMP位图压缩算法中的动态规划部分，通过计算每个像素点采用不同压缩方式所需要的比特数，得到最终的压缩方案。**

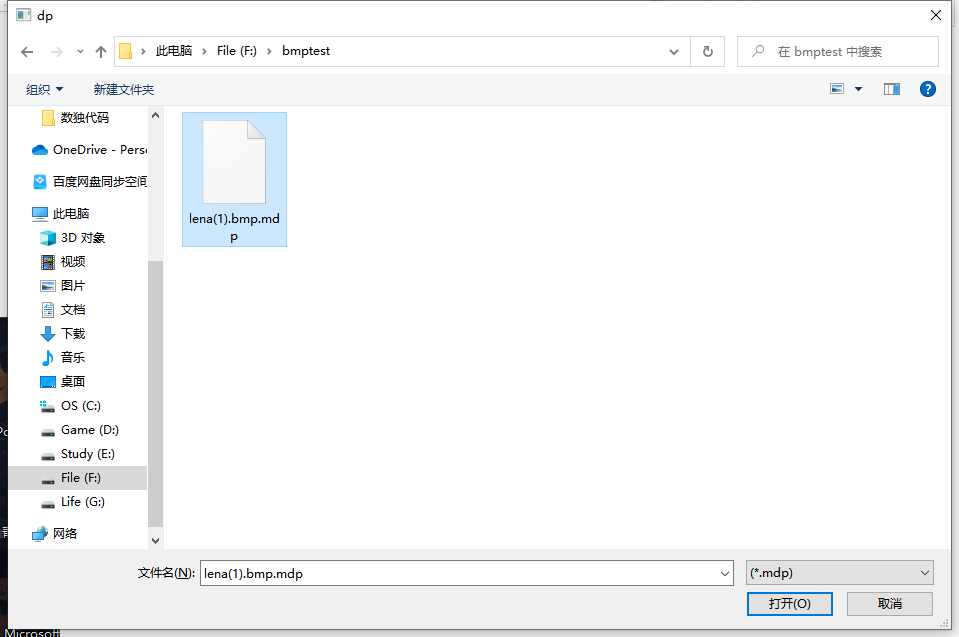
1. **功能测试与改进**

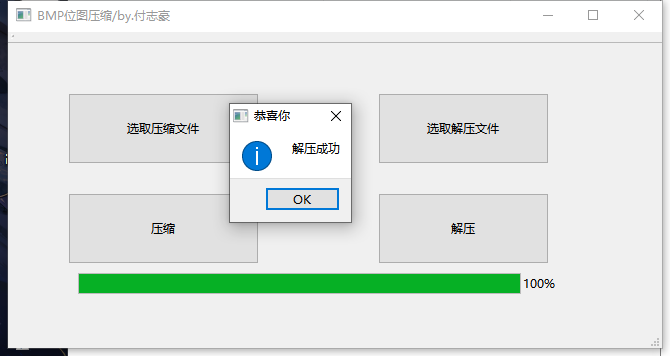
**对于功能测试：**

**功能主界面：**

**我们选取压缩文件：**

**在文件中会出现：****的压缩文件**

**我们对解压文件进行测试：**



在解压之后会出现一个新的文件：

对于改进：我预想可以在界面添加一个预览功能，解压缩之后的图片直接出现在我们软件的旁边。

**三、实习小结**

通过这一次的实习，我的编程能力得到了进一步的提升。对于bmp图像的一个读取，刚开始的时候这个题目我的思绪一点都没有。不知道改从何处下手，但是随着我做不断地深入下去去做，也会清晰很多。主要是刚开始的规划不清楚。对于每一步该怎么去做不是很清楚。然后我们主要是根据需求去做相应的一块一块就很好。

本次的QT学习也学习了不少知识，应该多多积累，本次对于图像的读取也有了更深一步的学习，能够知道图像在计算机之中是如何储存。动态规划算法中，我首先计算了二进制像素段P存储的位数，并确定了每个像素段的压缩方案，以实现最小化压缩率的目标。在追溯解中，我采取了非递归的方法，避免了数据量过大导致栈溢出的问题，使程序的运行效率更高。

使我对动态规划的理解更加深刻，只要我们不断努力，我们就可以解决问题。

-------------------------------

成绩评定：

教师签名：

批改日期：