实验报告

1. 问题描述

利用Huffman树和Huffman编码，完成文件的压缩和解压缩

## （2） Huffman压缩算法原理

先通过对要压缩的文件进行统计频率，比如在"65da as 美65a"中a-3（表示a出现3次），d-1，s-1，6-2，5-2，美-1，空格-2。Huffman采用不定长进行存储，频率高的对应的编码长度较短，频率低的对应的编码长度较长。

## （3） 创立Huffman编码表

1)先对压缩文件的字符进行频率统计，以"字符--频率"的形式存入某容器m

2) 在容器m中取出两个频率最小对应的字符，作为二叉树的两个叶子节点，并将频率和作为它们的根节点，同时将新结点存入容器m，将旧的两个结点踢出容器m。（容器m可以是优先队列）

3)重复(2)，直到最后容器m中只有一个元素。

4) 将形成的二叉树的左节点标0，右节点标1。把从最上面的根节点到最下面的叶子节点途中遇到的0,1序列串起来，就得到了各个符号的编码。

**例子：有一串"cdbedfaabca"，进行Huffman编码和解码。**

编码：

1）频率统计  f:1   e:1   d:2   c:2   b:2   a:5

2）f与e作为叶子结点，其根节点为\_2 。 此时，新的频率表为：\_2   d:2   c:2   b:2   a:5

d与\_2作为叶子结点，其根节点为\_4 。此时，新的频率表为：c:2   b:2   \_4   a:5

b与  c作为叶子结点，其根节点为\_4 。此时，新的频率表为：\_4   \_4   a:5

  \_4与\_4作为叶子结点，其根节点为\_8 。此时，新的频率表为： \_8   a:5

    \_8与  a作为叶子结点，其根节点为\_13 。

    结束。

3）左子树标0，右子树标1。如下图所示：

Huffman编码表：a:0  c:100  b:101  f:1100  e:1101  d:111

 "cdbedfaabca"的编码"10011110111011111100001011000"

解码：

读取第一位‘1’，搜索Huffman表，找不到。继续读下一位"10"，找不到，继续读下一位"100"，此时对应字符‘c’。清零，继续读第一位"1"，直至读到"111"，对应‘d’。继续清零，继续读第一位‘0’，直至读到"101"，对应‘b’。... ... 读到最后，得"cdbedfaabca"。

**进行Huffman编码和译码时，避免使用文本文件来读写，采用二进制文件。在读取原文件和输出译码后的文件时，用文本文件，而不用二进制文件，这样才能保证文件打开时能正常显示换行。**

## （4） 压缩算法，算法仅供参考

a) 统计一个字节值[0,255]的出现频度：一维数组存储频度。对应下标元素的频率统计值。

b) 构建哈夫曼树：

需要把频度大的节点放在离根近的地方，频度小的节点放在离根远的地方。所以，需要进行最小字符频度的查找。在没有访问过的节点中，找最小字符频度下标来构建哈夫曼树。

c) 构建哈夫曼编码：可以采用递归构建

d) 生成压缩文件：

把字节的哈夫曼编码以二进制形式写入目标文件中。给压缩文件头部写入元数据，解压缩时需使用这些数据。

把每个字符的哈夫曼编码一个一个过，如果是字符’0’，就转换为二进制的0；如果是字符’1’，就转换为二进制的1。转换结果从第1位依次赋值，若已经八位（一个字节）了，就写入文件中。如果最后一次不满一个字节，依然需要写到文件中，注意：写入的最后一个字节可能会存在垃圾位

注意：给目标文件写二进制数据时，最后一个字节如果不满八位，会产生垃圾数据，如果不进行处理，在解压后，就不能完整的还原。所以，需记录原始文件字节数，或者计算最后一个字节中的有效位数。

## （5） **解**压缩****过程，算法仅供参考

a) 获取Huffman码表

根据压缩文件的头部元数据，得到哈夫曼编码，构建哈夫曼树

b)  解码

生成解压缩的文件（后缀名和源文件后缀名一样 ）即可：一位一位的从压缩文件中读取信息，’1’向左子树走，’0’向右子树走。

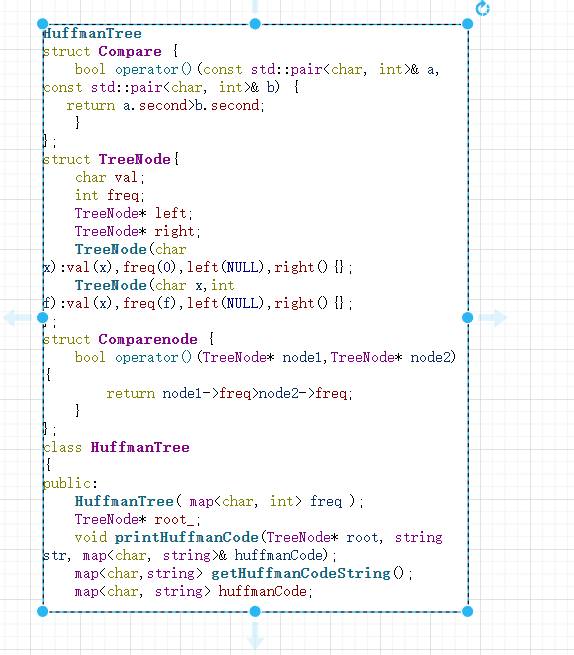
应该先利用fseek()函数把文件指针跳过前面的元数据和字符种类及频度，定位到真正需要还原的地方。取出的一个字节，一位一位的进行判断，’1’向左子树走，’0’向右子树走；若到达叶子结点，向文件中写入叶子结点下标对应的字符。再回到根结点继续；若超过一个字节，8位，则需要读取下一个字节。

压缩文件头部有元数据，所以，解压时需要把文件指针定位到真正的信息处。当碰到最后一字节的垃圾位时，应结束，否则解压出的信息和源文件会不匹配。

1. 设计思路

HuffmanTree：

UML类图：



关于如何构建哈夫曼树的，创建叶子节点，比较大小函数，获取哈夫曼编码，返回哈夫曼编码等等操作

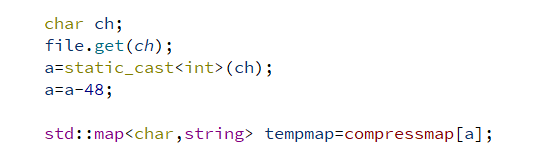


选择多个文件进行解压缩的关键步骤

For(const QString &fileName:fileNames)

将选中的文件存在QStringList当中，以for循环的方式逐个压缩和解压缩文件

关于如何实现多个压缩文件中选中一个压缩文件进行解压缩



对哈夫曼编码进行记忆性存储，每次对文件进行解压缩的时候，先获取它的哈夫曼编码，从而能够进行解压缩。

1. 测试即运行结果

