压缩软件

一、 问题描述

设计一个压缩软件，能对输入的**任何类型的文件**进行霍夫曼编码，产生编码后的压缩文件；也能对输入的压缩文件进行译码，生成压缩前的文件一解压文件。

二、基本要求

要求编码和译码的效率尽可能高，解压后的文件和压缩前相同。

三、工具及准备工作

硬件：联想 `ThinkBook 16+`

软件：VS 2022

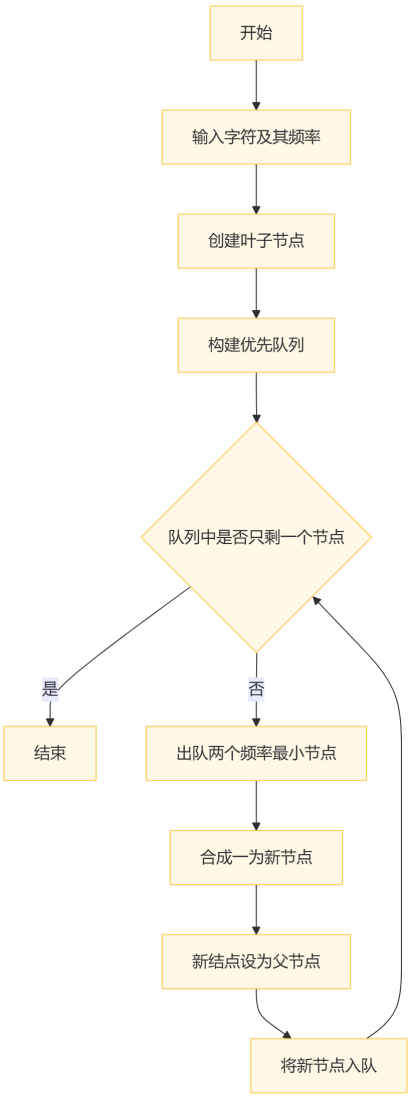
四、分析与实现

关键在于霍夫曼树模板类的实现，注意在编解码时，由于一进制位操作较困难，因此在霍夫曼树模版类实现时，以字符‘O’代表二进制位0，以字符‘1’代表二进制值位1，因此每一个待编字符对应着一个字符元素是‘0’和“1’的字符串，在压缩、解压时，需要将编码字符串转换为二进制位。

创建霍夫曼树时，生成每个待编字符的编码，放在一个表中，编码时查表就能得到待编字符的编码。

译码时，把二进制位序列转换为‘0’、‘1’字符序列，输入霍夫曼树进行译码，译码从当前译码位置开始(需要在模版类中定义当前译码节点，初始时指向根节点)，根据编码序列，‘0’走左分支，‘1”走右分支，当到达叶子节点时，得到译码对象，并把当前译码位置回归到根节点。

霍夫曼算法使用了贪心的思想，得到了最小的编码路径。



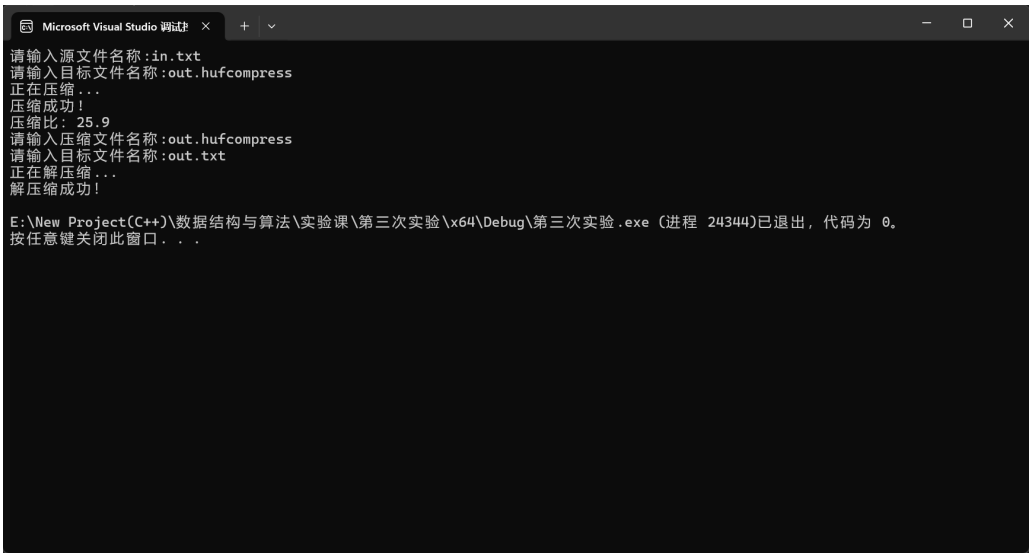
五、测试与结论

为了测试代码的通用性，分别压缩**纯文本文件**、**Word文档**、**图像文件**，然后再解压，查看压缩前文件和解压后的文件是否相同，通过实际测试结果反应本程序是否满足课程设计的基本要求。

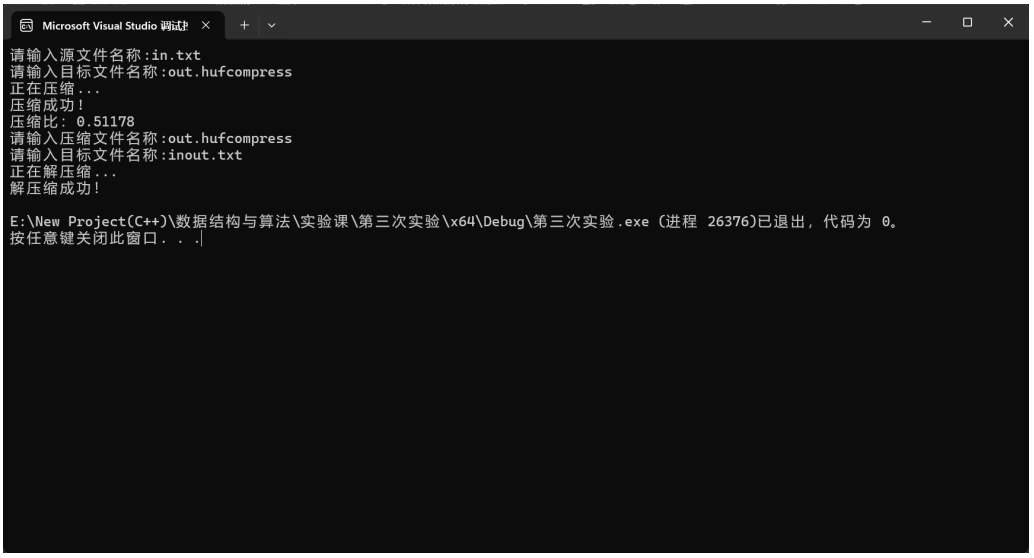
1. 纯文本文件

   测试发现，当纯文本文件较小时，压缩效果差（由于压缩文件前面的压缩信息大小固定），当纯文本文件大小增大时，压缩效果明显。

   - 小文本

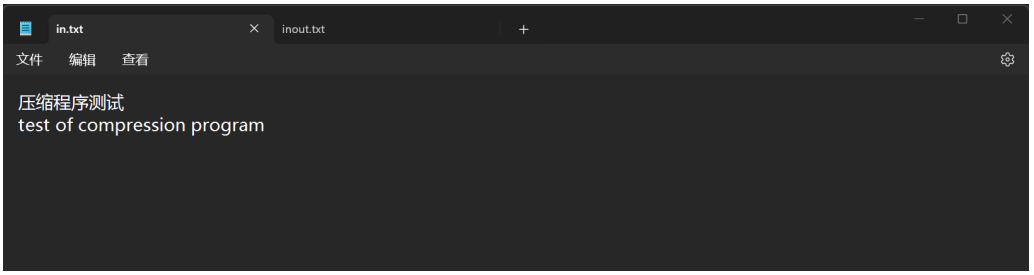


- 稍大文本：

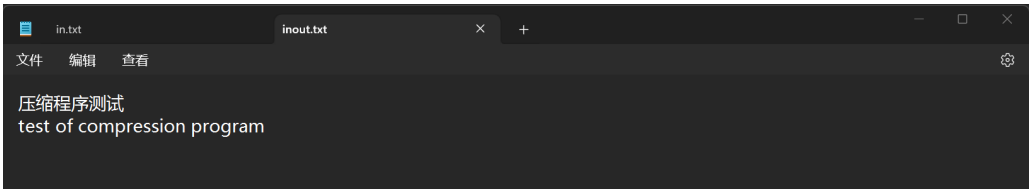


最后对比压缩前与压缩后的文本，结果相同

- 压缩前：

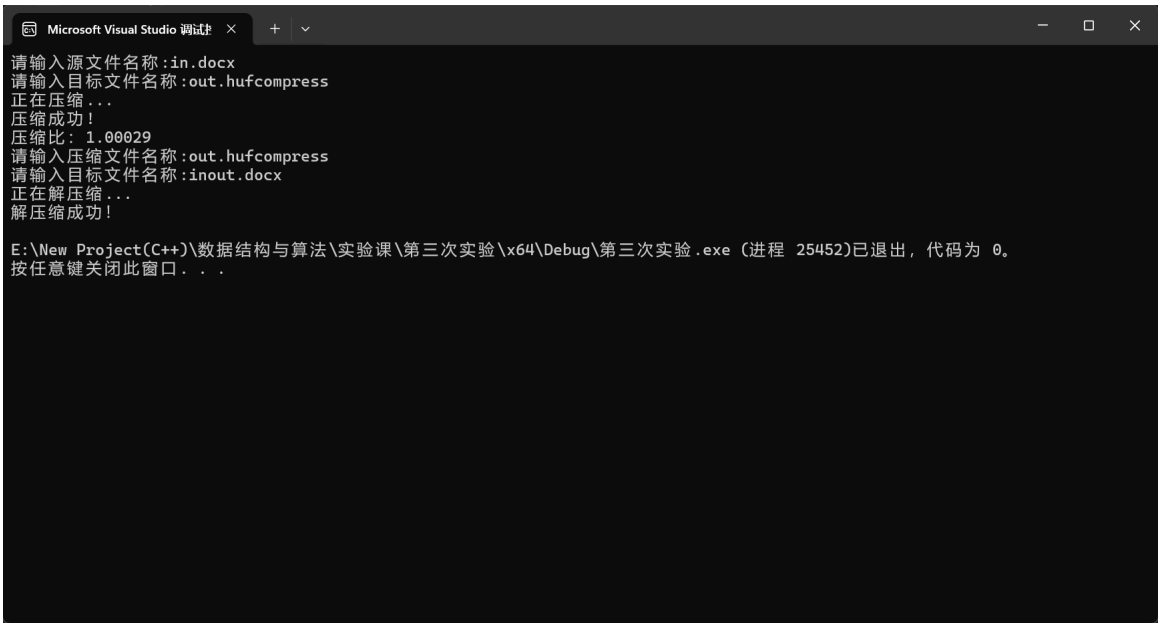


- 压缩后：



2. Word文件

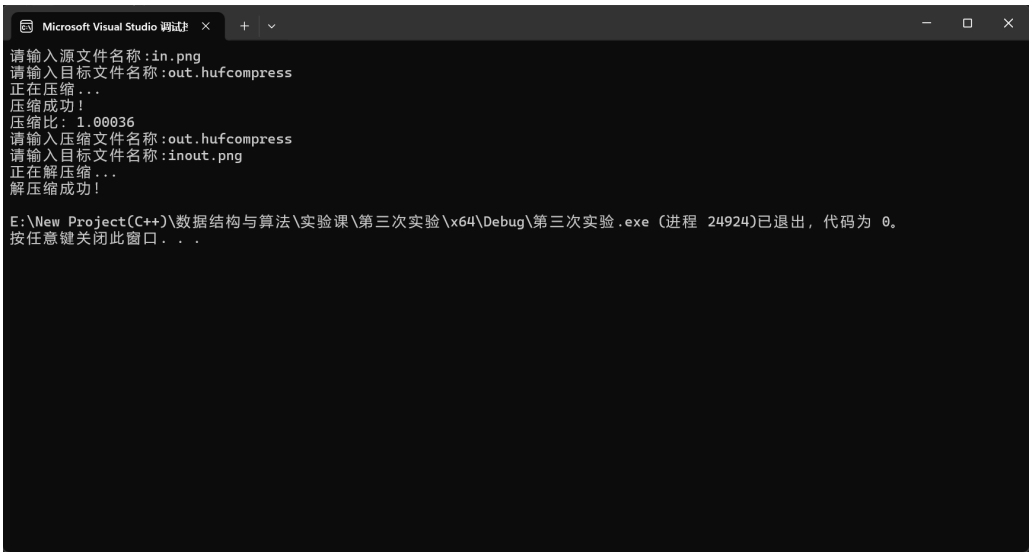
选择较大的复杂Word文档，通过压缩解压发现二者一致。



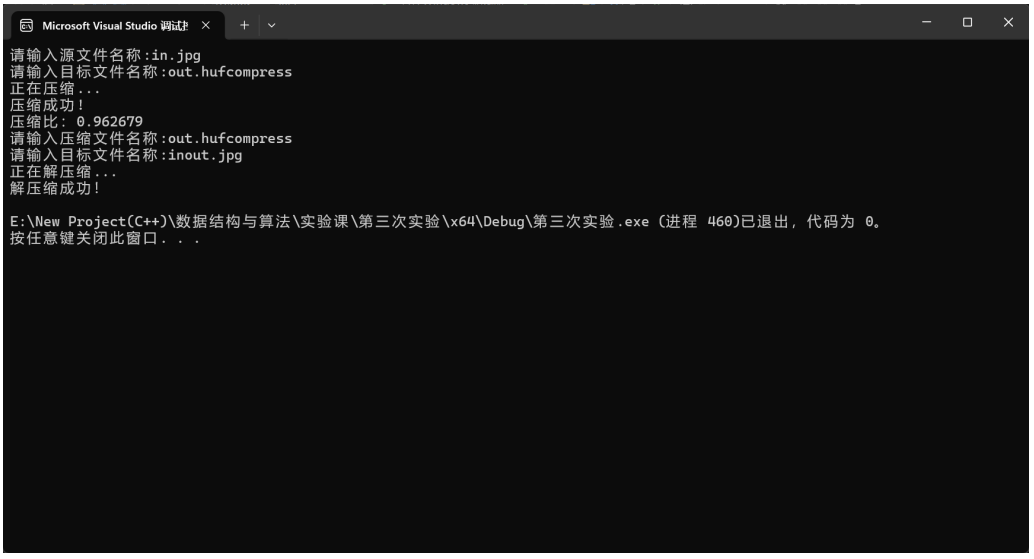
3. 图像文件

选择较大的复杂Word文档，通过压缩解压发现二者一致。

- PNG文件



- JPG文件



综上所述，压缩程序测试正常且具有良好的通用性，对基本上所有文件适用。

六、思考与感悟

1. 关于霍夫曼压缩程序的局限性

在实验过程中，我们观察到霍夫曼压缩程序对于Word文档、PNG图像和JPG图像等文件格式的压缩效果并不显著，压缩比大致维持在1左右。这一现象引发了我对文件压缩技术深层次的思考。这些文件格式本身可能已经采用了如LZ77、DEFLATE、或者特定的图像压缩算法（如PNG的过滤和DEFLATE组合，JPG的离散余弦变换和量化步骤）等高度优化的压缩技术。这些技术针对文件内容的特性进行了精细化的处理，从而在很大程度上消除了冗余信息，实现了高效的压缩。因此，当我们尝试使用霍夫曼压缩程序对已经经过高效压缩的文件进行二次压缩时，能够进一步压缩的空间自然变得非常有限。这一发现不仅让我意识到文件压缩技术的复杂性和多样性，也提醒我在实际应用中需要更加细致地考虑压缩算法的选择和适用性。

2. 关于git版本管理的深入理解

通过本次实验，我进一步熟悉了git这一强大的版本管理工具。在项目的迭代过程中，我深刻体会到了git在代码追踪、版本回退、分支管理等方面的便捷性。使用git，我能够轻松地记录每一次代码修改的历史，确保项目进展的每一步都可追溯。同时，git的分支功能使得我能够在不影响主线开发的情况下，进行新功能的尝试和bug的修复，极大地提高了开发效率和代码质量。这次实验让我更加深刻地认识到，掌握git等版本管理工具对于软件开发者来说是不可或缺的技能。

3. 对树形结构和哈夫曼算法的再认识

实验过程中，我再次深入接触到了树形结构和哈夫曼算法。树形结构作为一种重要的数据结构，在哈夫曼编码的生成过程中发挥着关键作用。通过构建哈夫曼树，我们能够根据字符出现的频率为其分配不同长度的编码，从而实现数据的压缩。这一过程不仅锻炼了我的逻辑思维能力，也加深了我对树形结构和哈夫曼算法原理的理解。我意识到，在实际应用中，合理地选择和使用数据结构对于优化算法性能、提高程序效率至关重要。

4. C++编程特性的感悟

通过本次实验，我更加深入地体会到了C++编程中模板、多态和继承等特性的便利性。模板的使用使得我能够编写出更加通用和灵活的代码，减少了重复劳动；多态的特性则允许我以统一的方式处理不同类型的对象，提高了代码的扩展性和可维护性；而继承则让我能够充分利用已有的类来构建新的类，实现了代码的重用和层次的划分。这些特性不仅提高了我的编程效率，也让我更加深刻地认识到了面向对象编程的魅力和优势。在未来的学习和工作中，我将继续深入探索C++的这些特性，努力提升自己的编程水平。