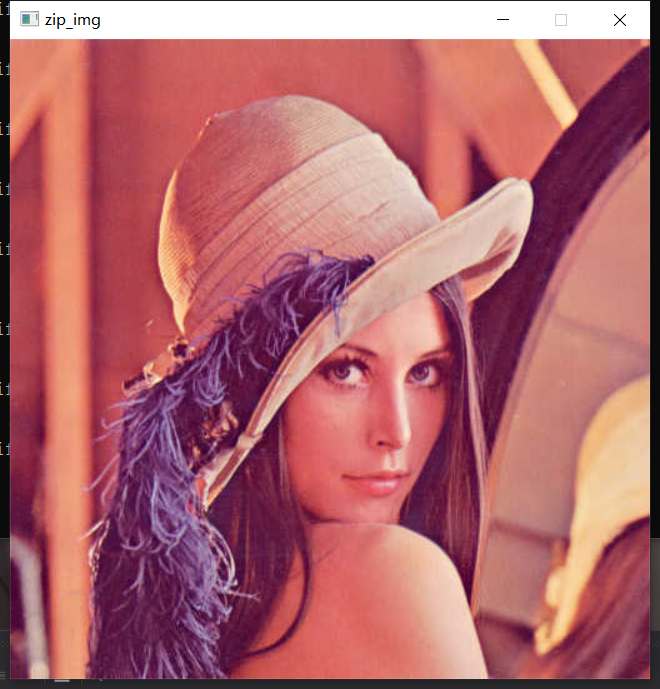
：

* 报告名称：

《图像压缩大作业实验报告》

zip\_img.exe -compress lena.tiff  
zip\_img.exe -read lena.jpg



* 专业/班级：济勤/16班
* 学号：2152957
* 姓名：罗国蔚
* 完成日期：2022/5/8

(2) 设计思路与功能描述

写出为Jpg形式

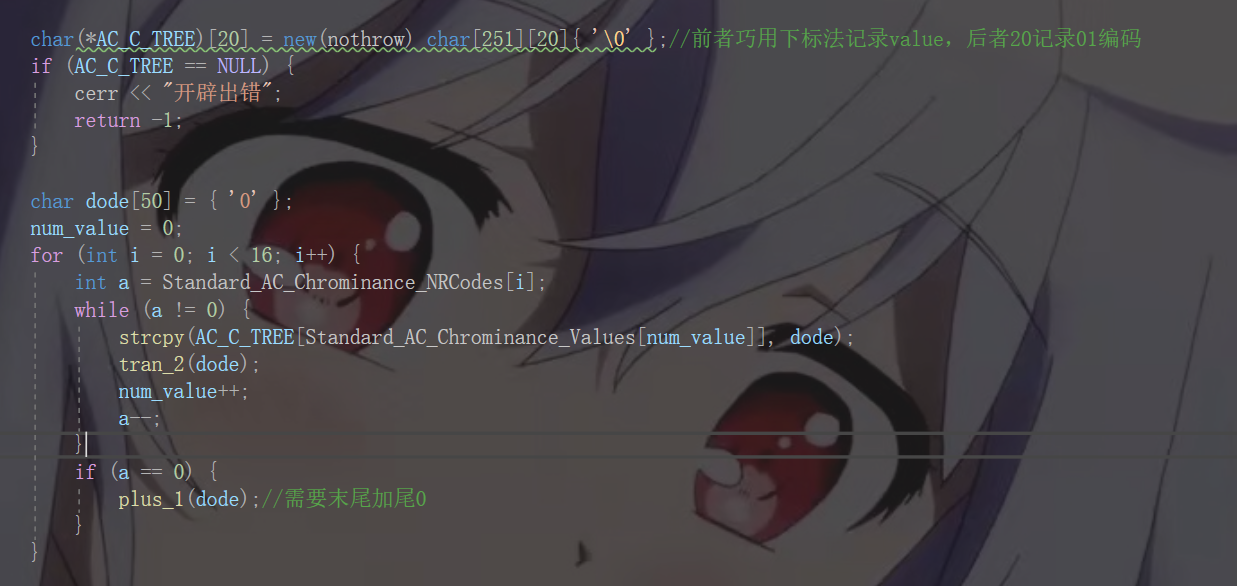
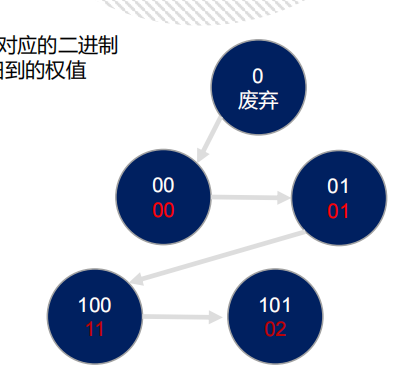
其实说实话，最开始根本就下不了手，感觉学长的ppt太难看懂了，但是后面发现，自己根本就没有思路写其他的，只能按照学长ppt按部就班，跟着室友一起写

首先是浏览了ppt但还是感觉一脸茫然，然后呆了几天，后面重新开动，开了压缩包里的picreader，才感觉慢慢有点头绪，在picreader里面就大约估计了一个图片的构成要素（虽然后续发现和jepg差距很大）

Step1

首先还是将文件放入联立，包括头文件啥的，然后构造main.cpp，然后同上一个文件压缩一样，分成两模块，通过主函数传参实现不同功能，（当然在上述过程中还建立了几个数组（jepg推荐数组））

Step2 -compress建树 部分

首先根据几个数组建立了四个Huffman树，（yysy，这树感觉比上一次的还烦，可能是因为要从上一次的建树规则跳入到这一次的范式Huffman有点难以接受，不过在室友的讲解之下还是很快就明白了建树的规则，而且在一次的建树过程中吸取经验，巧用数组下标达到对应标号的问题）通过一个初始字符‘0’，进行增加字符‘0’以及将0变成1的操作得到所谓的范式Huffman树（原理如图，代码语言也如图（选择其中一棵树作为代表））

Step3 -compress部分

仿照PIcReader获取tiff信息后，进入到写文件头（这个过程也很煎熬，主要是自己以前也几乎完全不懂一个文件的格式）后续还因为文件头写入有误导致打不开jpg，还一直在查其他地方的bug

Step4-compress算法核心部分

写完基本的文件头后就需要将整理好的ycbcr放入到文件中，而整理ycbcr三组数据则是该程序的核心了

Step4-1将图像分割为8\*8的小块，（并在8\*8循环外边开一个str类以存储每一次在8\*8内部获得的01序列串）在每一小块内部进行一系列操作如下

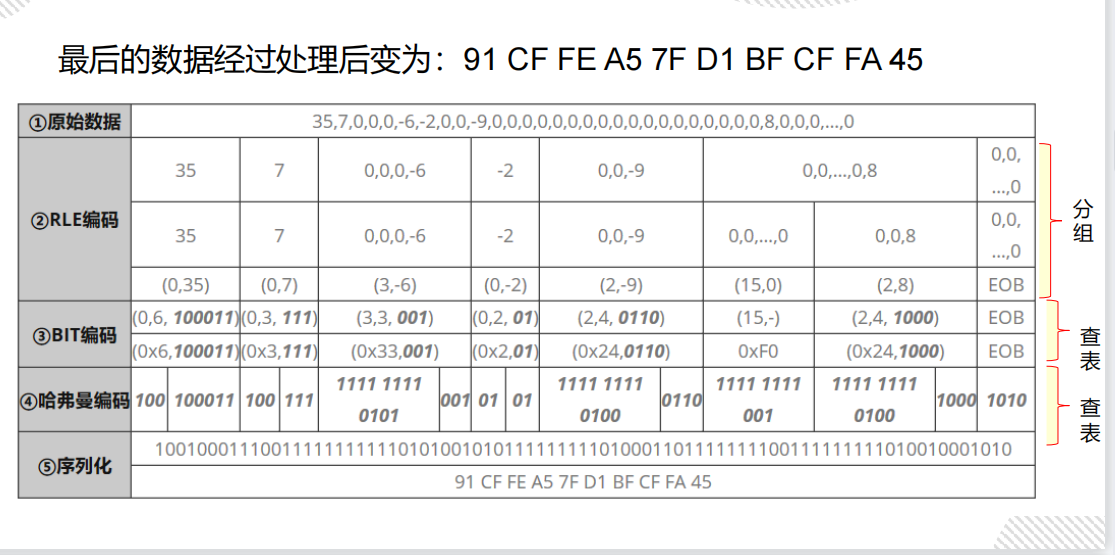
Step4-2 离散余弦变换

首先将获取到的rgba通过转化公式得到一组新的YCbCr，将得到的ycbcr数组经过变换公式存储到一个新的数组中，离散余弦变换的好处在于将很多数据集中到了数组前半段

Step4-3量化+zigzag

量化的计算步骤涉及到官方给的标准量化表，量化过后的数组就有很多0在后面了，此时需要zigzag进行z字抖动，使得0更加集中，而使0更加集中的好处在于后续能用一个eob即可代替许多0的存储

Step4-4 rle编码加Huffman查表

本人是将两个编码揉在了一个较大的循环里面，感觉因为条件太多，感觉很shi山，具体思路见下图（顺序如箭头所示）

①将原始数据通过循环且判断是否为零分成小组，一个数字记录前面0的个数，一个数字记录非0数的个数，并在有零时单独开一个“向后窥探”的一个循环，看是否满足是eob，②然后通过几个有关二进制和str的函数将35转化为一个二进制字符串，如果35＜0，则将01字符串反序一下，③得到的长度和前者0的个数组合得到一个16进制数字带入Huffman树中去找数，将对应的字符串先放入一个稍小的字符数组中，后续将二进制也放入，最后将该数组拷贝到循环外的一个str中，至此核心部分就结束了（but这个过程真的很搅，心态很炸）

Step5将str中的数据放入到文件中，此时又有许多小坑的地方，比如y省的几个01要添在cb前，cb省的同理加在y之前，最后通过一个循环算法，将每八个放入到一个数字中，（并且如果是ff时候要在后面跟00，小坑，最后还被坑的地方在于自己写入的是一个un char 型，在做判断的时候却用了==char（0xff）导致最后没有达到预期，这个小bug也是想了半天）

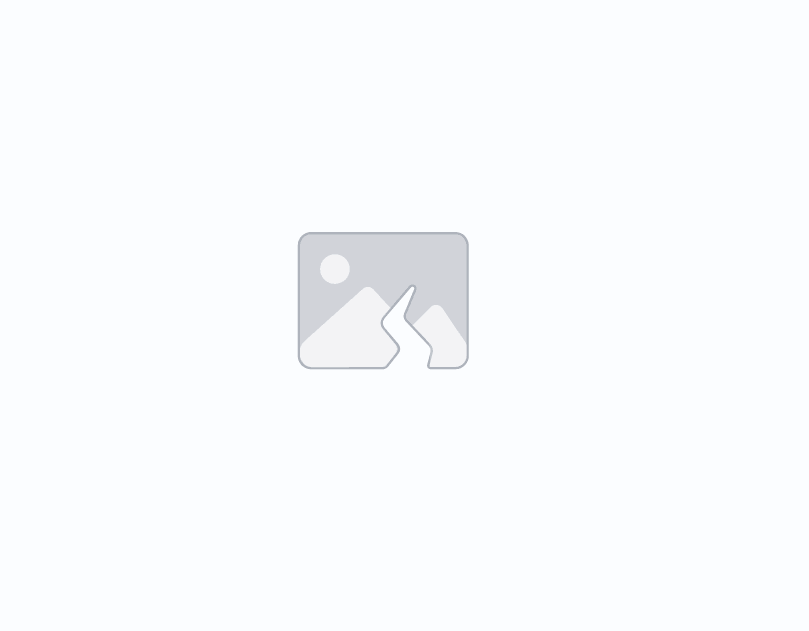
(3) 在实验过程中遇到的问题及解决方法

①建树问题

建树这有两个问题吧一个是建树本身的思维，另一个是开辟动态数组的问题，

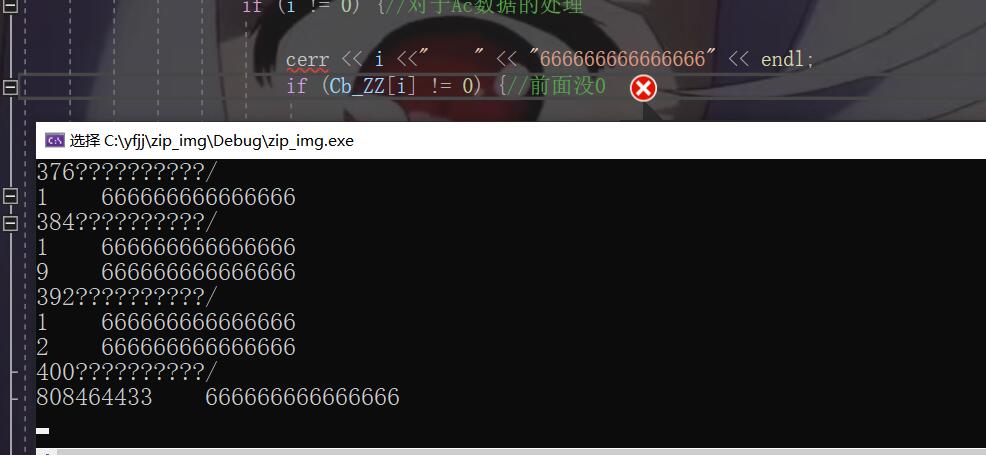
建树本身的问题最开是看ppt完全就没懂，后续在室友的教育下才理解逻辑，是直接拿官方给的标准数组直接开始建树，通过0的增加和01换得到编码串，第二动态开辟的问题，最初是因为开的静态数组，但是后面越界报错就查了很久，最后想到可能是栈爆了的原因，果然main下面给了报错，后面改为动态，不过因为数组开小了应该是251\*20写成了250\*20，导致后续越界，也想了半天，不过这里debug让我明白了调试报×的一行并不一定是问题本身原因，可能在上面就出现了问题，只不过是在这一行表现了出来，然后也更加懂了数组越界的debug其实是很难找出来的，而且报错形式都可能有多种

②写文件头问题

最开始根本没理解“就嗯写”的意思，后面懂了之后也还是输入理解上有问题，主要是对x字节的输入有问题比如2字节的数（原本我开始以为是out<<short(i)）后续才知道自己经过运算拆成两个char输进去。然后头文件有个地方有个小错误导致后续jpg格式打不开，还找了大半天吧（因为肉眼搜索一直感觉是对的），后来的解决方法是直接看代码里面的输入输出的这就是打不开的图，（不过好在最后一次成功的时候就是一次到位，没有花花绿绿的）

③rle加Huffman问题

Rle+huffman一共有两个问题，第一个是二进制的0的长度，在写代码的时候我就发现好像如果是这样编码0和-1根本区别，二进制码都是0，长度又都是0，后来问室友才知道0的二进制长度直接算0，然后就解决了，第二个是rle中有超过16个0之后后续还有非零数的时候就会有bug出现，解决的方法是如果已经连续出现16个0，直接eob，直接不管后续有没有非零数了（因为感觉如果后续只是丢了一点点数据并不影响肉眼观察），后续证明查看图片确实没有问题。

④大循环卡死问题

其实这个的debug是最摸不着头脑的，问了好些人也没有得到什么实质性的进展，最开始从这发现是栈爆炸了，后续得知str不会占用栈然后就迷了，后面de是发现是因为有数组越界带来的问题，然后解决rle的16个0的eob问题就自然就好了

1. 心得体会

说实话这次大作业感触颇多，成长了很多

①debug的路只有自己走，室友最多能帮到你一些地方，自己的bug是只有靠自己慢慢推敲的，别人最多给你解决一些思路或者硬性要求上的问题

②自己真的要养好心态，不要轻易放弃，说实话，写这份大作业的时候一开始就想放弃，写到一半也想放弃，写到最后de了很久也想放弃，中途甚至因为坐的太久已经开始跪着写程序了，不过最终取得了阶段性成果，一切看来是心酸的又是值得的

③任何东西只要努力了就有希望，当最初ddl日来临的时候我原本以为我输出到文件中就没啥问题的时候结果循环炸了（因为之前每一步几乎都是打印了出来和室友的都是一样的，就以为能一遍能成，而且他们也帮我踩了不少坑，dc的偏移值，ff后+00等），就当我以为赶不上ddl的时候大作业就以为延期两天，当时就觉得无论如何也得把这个作业写完整，真的特别感谢老师和TA的延期两天救了孩子的命

1. 源代码

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "PicReader.h"

#include <stdio.h>

#include <stdlib.h>

#include <fstream>

#include <iostream>

#include <math.h>

#include <string>

#define PI 3.14159

using namespace std;

void round\_f(double org[64], int Q[64], int round\_[64])

{

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

double r = org[8 \* i + j] / Q[8 \* i + j];

round\_[8 \* i + j] = int(round(r));

}

}

}

double alpha(int i)

{

if (i == 0) {

return (1.0 / sqrt(8));

}

else

return 1.0 / 2;

}

void tran\_2(char p[])

{

int length = strlen(p);

//从右往左进行运算改变（拟实现2进制运算）

for (int i = length; i > -1; i--) {

if (p[i] == '0') {

p[i] = '1';

break;

}

if (p[i] == '1') {

p[i] = '0';

}

}

}//进行2进制加法模拟

void plus\_1(char p[])

{

int length = strlen(p);

//cerr << "8"<<" "<<length << endl;

p[length] = '0';

p[length+1] = '\0';

}//在后续添加0

void str\_fan(char p[], int length)

{

for (int i = 0; i < length / 2; i++) {

char a = p[i];

p[i] = p[length - i - 1];

p[length - i - 1] = a;

}

}

void length\_2\_f(char p[],int n)

{

int i = 0;

if (n < 0) {

n = -n;

while ((n / 2) != 0)//将十进制数不断除2取余数，直到最后除2等于0，循环结束

{

p[i] = n % 2+48;//将十进制数除2后的余数存放在数组中

i++;

n /= 2;//不断改变n的值

}

p[i] = n+48;

return;

}

if (n > 0) {

while ((n / 2) != 0)//将十进制数不断除2取余数，直到最后除2等于0，循环结束

{

p[i] = n % 2+48;//将十进制数除2后的余数存放在数组中

i++;

n /= 2;//不断改变x的值

}

p[i] = n+48;

return;

}

if (n == 0) {

p[0] = '0';

}

}

const char Standard\_DC\_Luminance\_NRCodes[] = { 0, 0, 7, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0 };//亮度Y

const unsigned char Standard\_DC\_Luminance\_Values[] = { 4, 5, 3, 2, 6, 1, 0, 7, 8, 9, 10, 11 };

//-------------------------------------------------------------------------------

const char Standard\_DC\_Chrominance\_NRCodes[] = { 0, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 };//C

const unsigned char Standard\_DC\_Chrominance\_Values[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };

//-------------------------------------------------------------------------------

const char Standard\_AC\_Luminance\_NRCodes[] = { 0, 2, 1, 3, 3, 2, 4, 3, 5, 5, 4, 4, 0, 0, 1, 0x7d };

const unsigned char Standard\_AC\_Luminance\_Values[] =

{

0x01, 0x02, 0x03, 0x00, 0x04, 0x11, 0x05, 0x12,

0x21, 0x31, 0x41, 0x06, 0x13, 0x51, 0x61, 0x07,

0x22, 0x71, 0x14, 0x32, 0x81, 0x91, 0xa1, 0x08,

0x23, 0x42, 0xb1, 0xc1, 0x15, 0x52, 0xd1, 0xf0,

0x24, 0x33, 0x62, 0x72, 0x82, 0x09, 0x0a, 0x16,

0x17, 0x18, 0x19, 0x1a, 0x25, 0x26, 0x27, 0x28,

0x29, 0x2a, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,

0x3a, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49,

0x4a, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59,

0x5a, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69,

0x6a, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79,

0x7a, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89,

0x8a, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98,

0x99, 0x9a, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7,

0xa8, 0xa9, 0xaa, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6,

0xb7, 0xb8, 0xb9, 0xba, 0xc2, 0xc3, 0xc4, 0xc5,

0xc6, 0xc7, 0xc8, 0xc9, 0xca, 0xd2, 0xd3, 0xd4,

0xd5, 0xd6, 0xd7, 0xd8, 0xd9, 0xda, 0xe1, 0xe2,

0xe3, 0xe4, 0xe5, 0xe6, 0xe7, 0xe8, 0xe9, 0xea,

0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8,

0xf9, 0xfa

};

//-------------------------------------------------------------------------------

const char Standard\_AC\_Chrominance\_NRCodes[] = { 0, 2, 1, 2, 4, 4, 3, 4, 7, 5, 4, 4, 0, 1, 2, 0x77 };

const unsigned char Standard\_AC\_Chrominance\_Values[] =

{

0x00, 0x01, 0x02, 0x03, 0x11, 0x04, 0x05, 0x21,

0x31, 0x06, 0x12, 0x41, 0x51, 0x07, 0x61, 0x71,

0x13, 0x22, 0x32, 0x81, 0x08, 0x14, 0x42, 0x91,

0xa1, 0xb1, 0xc1, 0x09, 0x23, 0x33, 0x52, 0xf0,

0x15, 0x62, 0x72, 0xd1, 0x0a, 0x16, 0x24, 0x34,

0xe1, 0x25, 0xf1, 0x17, 0x18, 0x19, 0x1a, 0x26,

0x27, 0x28, 0x29, 0x2a, 0x35, 0x36, 0x37, 0x38,

0x39, 0x3a, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,

0x49, 0x4a, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58,

0x59, 0x5a, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68,

0x69, 0x6a, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78,

0x79, 0x7a, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,

0x88, 0x89, 0x8a, 0x92, 0x93, 0x94, 0x95, 0x96,

0x97, 0x98, 0x99, 0x9a, 0xa2, 0xa3, 0xa4, 0xa5,

0xa6, 0xa7, 0xa8, 0xa9, 0xaa, 0xb2, 0xb3, 0xb4,

0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xba, 0xc2, 0xc3,

0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xca, 0xd2,

0xd3, 0xd4, 0xd5, 0xd6, 0xd7, 0xd8, 0xd9, 0xda,

0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7, 0xe8, 0xe9,

0xea, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8,

0xf9, 0xfa

};

int QY[64] = { 16,11,10,16,24,40,51,61,

12,12,14,19,26,58,60,55,

14,13,16,24,40,57,69,56,

14,17,22,29,51,87,80,62,

18,22,37,56,68,109,103,77,

24,35,55,64,81,104,113,92,

49,64,78,87,103,121,120,101,

72,92,95,98,112,100,103,99 };

int QC[64] = { 17,18,24,47,99,99,99,99,

18,21,26,66,99,99,99,99,

24,26,56,99,99,99,99,99,

47,66,99,99,99,99,99,99,

99,99,99,99,99,99,99,99,

99,99,99,99,99,99,99,99,

99,99,99,99,99,99,99,99,

99,99,99,99,99,99,99,99 };

const char ZigZag[64] =

{

0, 1, 5, 6,14,15,27,28,

2, 4, 7,13,16,26,29,42,

3, 8,12,17,25,30,41,43,

9,11,18,24,31,40,44,53,

10,19,23,32,39,45,52,54,

20,22,33,38,46,51,55,60,

21,34,37,47,50,56,59,61,

35,36,48,49,57,58,62,63

};

int main(int argc, char\* argv[]) {

///cerr << "\*\*";

if (argc != 3) {

cerr << "Please make sure the number of parameters is correct." << endl;

return -1;

}

if (!strcmp(argv[1], "-compress")) {

//进行压缩步骤

//huffman过程

int num\_value = 0;

/\*建AC\_C的树\*/

char(\*AC\_C\_TREE)[20] = new(nothrow) char[251][20]{ '\0' };//前者巧用下标法记录value，后者20记录01编码

if (AC\_C\_TREE == NULL) {

cerr << "开辟出错";

return -1;

}

char dode[50] = { '0' };

num\_value = 0;

for (int i = 0; i < 16; i++) {

int a = Standard\_AC\_Chrominance\_NRCodes[i];

while (a != 0) {

strcpy(AC\_C\_TREE[Standard\_AC\_Chrominance\_Values[num\_value]], dode);

tran\_2(dode);

num\_value++;

a--;

}

if (a == 0) {

plus\_1(dode);//需要末尾加尾0

}

}

/\*建AC\_Y的树\*/

char(\*AC\_Y\_TREE)[25] = new(nothrow) char[251][25]{ '\0' };

//cerr << "44444";

if (AC\_Y\_TREE[0] == NULL) {

cerr << "开辟出错";

return -1;

}

char bode[50] = { '0' };

num\_value = 0;

for (int i = 0; i < 16; i++) {

int a = Standard\_AC\_Luminance\_NRCodes[i];

while (a != 0) {

strcpy(AC\_Y\_TREE[Standard\_AC\_Luminance\_Values[num\_value]], bode);

tran\_2(bode);

num\_value++;

a--;

}

if (a == 0) {

plus\_1(bode);//需要末尾加尾0

}

}

/\*DC-Y的树\*/

char DC\_Y\_TREE[12][16] ;//前者巧用下标法记录value，后者16记录01编码

char code[16] = {'0'};

num\_value = 0;

for (int i = 0; i < 12; i++) {

int a = Standard\_DC\_Luminance\_NRCodes[i];

//cerr << "0" << endl;

while (a !=0) {

strcpy(DC\_Y\_TREE[Standard\_DC\_Luminance\_Values[num\_value]], code);

tran\_2(code);

num\_value++;

a--;

}

if (a == 0) {

plus\_1(code);//需要末尾加尾0

}

}

/\*建DC\_C的树\*/

char DC\_C\_TREE[12][16];//前者巧用下标法记录value，后者16记录01编码

char aode[16] = { '0' };

num\_value = 0;

for (int i = 0; i < 12; i++) {

int a = Standard\_DC\_Chrominance\_NRCodes[i];

//cerr << "0" << endl;

while (a != 0) {

strcpy(DC\_C\_TREE[Standard\_DC\_Chrominance\_Values[num\_value]], aode);

tran\_2(aode);

num\_value++;

a--;

}

if (a == 0) {

plus\_1(aode);//需要末尾加尾0

}

}

cerr << "begin"<<endl;

PicReader imread;

BYTE\* data = nullptr;

UINT x, y;

imread.readPic(argv[2]);

imread.getData(data, x, y);

//x是列，y是行

/\*写文件头\*/

ofstream out("lena.jpg", ios::binary);

if (!out.is\_open()) {

cout << "创建文件失败" << endl;

return -1;

}

out << char(0xff)<< char(0xd8) << char(0xff) << char(0xe0) << char(0x00)<< char(0x10) << char(0x4A) << char(0x46) << char(0x49) << char(0x46) << char(0x00);

out << char(0x01) << char(0x01) << char(0);//??

out << char(0x00)<< char(0x01)<< char(0x00) << char(0x01) << char(0) << char(0);

//打表

out << char(0xFF)<< char(0xdb) << char(0x00)<< char(0x84);

out << char(0x00);

//zz型输出

int Q\_Y[64], Q\_C[64];

for (int i = 0; i < 64; i++) {

Q\_Y[ZigZag[i]] = QY[i];

Q\_C[ZigZag[i]] = QC[i];

}

for (int i = 0; i < 64; i++) {

out << char(Q\_Y[i]);

}

out << char(0x01);

for (int i = 0; i < 64; i++) {

out << char(Q\_C[i]);

}

//开始扫描

out << char(0xFF)<< char(0xC0) << char(0x00)<< char(0x11) << char(0x08);

out << char(0x02) << char(0x00)<< char(0x02)<< char(0x00);//图像高宽512的hex为200

out << char(0x03)<< char(0x01)<< char(0x11) << char(0x00) << char(0x02) << char(0x11) << char(0x01) << char(0x03) << char(0x11) << char(0x01);

//写Huffman树

out << char(0xFF) << char(0xC4)<< char(0x01)<<char(0xA2);

//111111直流亮度huffman树

out<< char(0x00);

//直流0 huffman表

for (int i = 0; i < 16; i++) {

out << char(Standard\_DC\_Luminance\_NRCodes[i]);

}

for (int i = 0; i < 12; i++) {

out << char(Standard\_DC\_Luminance\_Values[i]);

}

//22222交流亮度huffman树

out<<char(0x10);

//交流1 huffmanID 0

for (int i = 0; i < 16; i++){

out<< char(Standard\_AC\_Luminance\_NRCodes[i]);

}

for(int i = 0; i < 162; i++) {

out<< char(Standard\_AC\_Luminance\_Values[i]);

}

//33333直流色度huffman树

out<< char(0x01);

//直流0 huffmanID 1

for (int i = 0; i < 16; i++) {

out << char(Standard\_DC\_Chrominance\_NRCodes[i]);

}

for (int i = 0; i < 12; i++) {

out << char(Standard\_DC\_Chrominance\_Values[i]);

}

//44444交流色度huffman树

out << char(0x11);

//交流1 huffmanID 1

for (int i = 0; i < 16; i++) {

out << char(Standard\_AC\_Chrominance\_NRCodes[i]);

}

for (int i = 0; i < 162; i++) {

out <<char(Standard\_AC\_Chrominance\_Values[i]);

}

out << char(0xFF) << char(0xDA) << char(0x00) << char(0x0C) << char(0x03);

out<< char(0x01)<< char(0x00)<< char(0x02)<< char(0x11)<< char(0x03)<< char(0x11);

out << char(0x00) << char(0x3F) << char(0x00);

/\*写文件头结束\*/

string str;

int Y\_DC\_LAST = 0, CB\_DC\_LAST = 0, CR\_DC\_LAST = 0;

/\*分区域小模块处理\*/

for (int p = 0; p < int(x); p += 8) {//列

for (int q = 0; q < int(y); q += 8) {//行(总的大循环)

/\*再对每一小块进行转换\*/

double img\_Y[64] = { 0 };

double img\_Cb[64] = { 0 };

double img\_Cr[64] = { 0 };

int n = 0;

for (int a = p; a < p + 8; a++) {

for (int b = q; b < q + 8; b++) {

int i\_data = a\*x\*4+b\*4;

img\_Y[n] = 0.299 \* data[i\_data]+0.5870 \* data[i\_data+1]+0.114 \* data[i\_data+2] -128;

img\_Cb[n] = -0.1687 \* data[i\_data] -0.3313 \* data[i\_data + 1] + 0.50000 \* data[i\_data + 2];

img\_Cr[n]= 0.50000 \* data[i\_data] - 0.4187 \* data[i\_data + 1] - 0.0813 \* data[i\_data + 2];

n++;

}

}

double alpha\_u = 0, alpha\_v = 0;

double Y\_new[64] = { 0 }, Cb\_new[64] = { 0 }, Cr\_new[64] = { 0 };

for (int u = 0; u < 8; u++) {

for (int v = 0; v < 8; v++) {

alpha\_u = alpha(u);

alpha\_v = alpha(v);

double F\_Y = 0,F\_Cr=0,F\_Cb=0;

double sum\_Y = 0,sum\_Cr=0,sum\_Cb=0;

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

sum\_Y += img\_Y[8 \* i + j] \* cos((2 \* i + 1) \* u \* PI / 16) \* cos((2 \* j + 1) \* v \* PI/16);

sum\_Cr += img\_Cr[8 \* i + j] \* cos((2 \* i + 1) \* u \* PI / 16) \* cos((2 \* j + 1) \* v \* PI/16);

sum\_Cb += img\_Cb[8 \* i + j] \* cos((2 \* i + 1) \* u \* PI / 16) \* cos((2 \* j + 1) \* v \* PI/16);

}

}

F\_Y = alpha\_u \* alpha\_v \* sum\_Y;

F\_Cr = alpha\_u \* alpha\_v \* sum\_Cr;

F\_Cb = alpha\_u \* alpha\_v \* sum\_Cb;

Y\_new[8 \* u + v] = F\_Y;

Cb\_new[8 \* u + v] = F\_Cb;

Cr\_new[8 \* u + v] = F\_Cr;

}

}

int roundY[64] = { 0 },roundCb[64] = { 0 }, roundCr[64] = { 0 };

round\_f(Y\_new, QY, roundY);

round\_f(Cr\_new, QC, roundCr);

round\_f(Cb\_new, QC, roundCb);

//zigzag处理

int Y\_ZZ[64] = {0}, Cr\_ZZ[64] = {0}, Cb\_ZZ[64] = {0};

for (int j = 0; j < 64; j++) {

Y\_ZZ[int(ZigZag[j])] = roundY[j];

Cr\_ZZ[int(ZigZag[j])] = roundCr[j];

Cb\_ZZ[int(ZigZag[j])] = roundCb[j];

}

//RLE编码

char rle\_code\_y[1000] = { 0 }, rle\_code\_cr[1000] = { 0 }, rle\_code\_cb[1000] = { 0 };

int num\_16 = 0,num\_0=0,length\_2=0;//num\_0记录0的个数,length\_2记录字符串长度

//Y操作

for (int i = 0; i < 64; i++) {

num\_0 = 0, num\_16 = 0, length\_2 = 0;

char ycode[12] = { 0 };

if (i == 0) {//对于dc数据的特殊处理

int Y\_DC = Y\_ZZ[0] - Y\_DC\_LAST;//先作差

Y\_DC\_LAST = Y\_ZZ[0];//保留此次的下次需要做减法

if (Y\_DC == 0) {

strcpy(rle\_code\_y, "110");

}

else {

num\_0 = 0;

length\_2\_f(ycode, Y\_DC);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(ycode);//求出长度，解决负号

if (Y\_DC < 0) {

for (int n = 0; n < length\_2; n++) {

if (ycode[n] == '1')

ycode[n] = '0';

else

ycode[n] = '1';

}

}

str\_fan(ycode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_y, DC\_Y\_TREE[num\_16]);

strcat(rle\_code\_y, ycode);

}

}

if (i != 0) {//对于Ac数据的处理

if (Y\_ZZ[i] != 0) {//前面没0

length\_2\_f(ycode, Y\_ZZ[i]);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(ycode);

if (Y\_ZZ[i] < 0) {

for (int n = 0; n < length\_2; n++) {

if (ycode[n] == '1')

ycode[n] = '0';

else

ycode[n] = '1';

}

}

str\_fan(ycode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_y, AC\_Y\_TREE[num\_16]);

strcat(rle\_code\_y, ycode);

}

if (Y\_ZZ[i] == 0) {//前面有0

int n = i;

while (Y\_ZZ[n] == 0) {

num\_0++;

n++;

i++;

if (num\_0 >= 15)

break;

}//while结束后，指向一个非0数的

if (num\_0 >=15) {

strcat(rle\_code\_y, "1010");//y拷贝1010，c拷贝00

break;

}

else {

length\_2\_f(ycode, Y\_ZZ[i]);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(ycode);

if (Y\_ZZ[i] < 0) {

for (int n = 0; n < length\_2; n++) {

if (ycode[n] == '1')

ycode[n] = '0';

else

ycode[n] = '1';

}

}

str\_fan(ycode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_y, AC\_Y\_TREE[num\_16]);

strcat(rle\_code\_y, ycode);

}

}

}

}

//y结束

// cb的开始

for (int i = 0; i < 64; i++) {

num\_0 = 0, num\_16 = 0, length\_2 = 0;

char cbcode[12] = { 0 };

//cerr << CB\_DC\_LAST << " " << i << " " << Cb\_ZZ[0] << " " << Cb\_ZZ[1] << " " << Cb\_ZZ[2] << endl;

if (i == 0) {//对于dc数据的特殊处理

int cb\_DC = Cb\_ZZ[i] - CB\_DC\_LAST;//先作差

CB\_DC\_LAST = Cb\_ZZ[i];//保留此次的下次需要做减法

if (cb\_DC == 0) {

strcpy(rle\_code\_cb, "00");

}

else {

num\_0 = 0;

length\_2\_f(cbcode, cb\_DC);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(cbcode);//求出长度，解决负号

if (cb\_DC < 0) {

for (int n = 0; n < length\_2; n++) {

if (cbcode[n] == '1')

cbcode[n] = '0';

else

cbcode[n] = '1';

}

}

str\_fan(cbcode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_cb, DC\_C\_TREE[num\_16]);

strcat(rle\_code\_cb, cbcode);

}

}

if (i != 0) {//对于Ac数据的处理

if (Cb\_ZZ[i] != 0) {//前面没0

length\_2\_f(cbcode, Cb\_ZZ[i]);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(cbcode);

if (Cb\_ZZ[i] < 0) {

for (int n = 0; n < length\_2; n++) {

if (cbcode[n] == '1')

cbcode[n] = '0';

else

cbcode[n] = '1';

}

}

str\_fan(cbcode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_cb, AC\_C\_TREE[num\_16]);

strcat(rle\_code\_cb, cbcode);

}

if (Cb\_ZZ[i] == 0) {//前面有0

int n = i;

while (Cb\_ZZ[n] == 0) {

num\_0++;

n++;

i++;

if (num\_0 >=15)

break;

}//while结束后，指向一个非0数的

if (num\_0 >=15) {

strcat(rle\_code\_cb, "00");//y拷贝1010，c拷贝00

break;

}

else {

length\_2\_f(cbcode, Cb\_ZZ[i]);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(cbcode);

if (Cb\_ZZ[i] < 0) {

for (int n = 0; n < length\_2; n++) {

if (cbcode[n] == '1')

cbcode[n] = '0';

else

cbcode[n] = '1';

}

}

str\_fan(cbcode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_cb, AC\_C\_TREE[num\_16]);

strcat(rle\_code\_cb, cbcode);

}

}

}

}

//cb序列化结束

//cr开始

for (int i = 0; i < 64; i++) {

num\_0 = 0, num\_16 = 0, length\_2 = 0;

char crcode[12] = { 0 };

if (i == 0) {//对于dc数据的特殊处理

int CR\_DC = Cr\_ZZ[0] - CR\_DC\_LAST;//先作差

CR\_DC\_LAST = Cr\_ZZ[0];//保留此次的下次需要做减法

if (CR\_DC == 0) {

strcpy(rle\_code\_cr, "00");

}

else {

num\_0 = 0;

length\_2\_f(crcode, CR\_DC);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(crcode);//求出长度，解决负号

if (CR\_DC < 0) {

for (int n = 0; n < length\_2; n++) {

if (crcode[n] == '1')

crcode[n] = '0';

else

crcode[n] = '1';

}

}

str\_fan(crcode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_cr, DC\_C\_TREE[num\_16]);

strcat(rle\_code\_cr, crcode);

}

}

if (i != 0) {//对于Ac数据的处理

if (Cr\_ZZ[i] != 0) {//前面没0

length\_2\_f(crcode, Cr\_ZZ[i]);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(crcode);

if (Cr\_ZZ[i] < 0) {

for (int n = 0; n < length\_2; n++) {

if (crcode[n] == '1')

crcode[n] = '0';

else

crcode[n] = '1';

}

}

str\_fan(crcode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_cr, AC\_C\_TREE[num\_16]);

strcat(rle\_code\_cr, crcode);

}

if (Cr\_ZZ[i] == 0) {//前面有0

int n = i;

while (Cr\_ZZ[n] == 0) {

num\_0++;

n++;

i++;

if (num\_0>=15)

break;

}//while结束后，指向一个非0数的

if (num\_0 >=15) {

strcat(rle\_code\_cr, "00");//y拷贝1010，c拷贝00

break;

}

else {

length\_2\_f(crcode, Cr\_ZZ[i]);//现在的ycode是反的//若Y\_DC是负数还需要换01；

length\_2 = strlen(crcode);

if (Cr\_ZZ[i] < 0) {

for (int n = 0; n < length\_2; n++) {

if (crcode[n] == '1')

crcode[n] = '0';

else

crcode[n] = '1';

}

}

str\_fan(crcode, length\_2);

num\_16 = num\_0 \* 16 + length\_2;

strcat(rle\_code\_cr, AC\_C\_TREE[num\_16]);

strcat(rle\_code\_cr, crcode);

}

}

}

}

//cr结束

str += rle\_code\_y;

str += rle\_code\_cb;

str += rle\_code\_cr;

}

}//至此是8\*8小块操作完毕

for (unsigned int i = 0; i < str.length();) {

unsigned char a = 0, c = 0;

for (unsigned int m = i,n=0; m < i+8; m++) {

if (str[m] != '\0') {

a = 0;

a |= ((str[m] - '0') << (7 - n));

c |= a;

n++;

}

if (m >= str.length())

break;

}

out << c;

if (c == unsigned char(0xff)) {

out << char(0x00);

}

i += 8;

if (i >= str.length() - 1) {

break;

}

}

out <<char(0xFF) <<char(0xD9);

delete[] AC\_C\_TREE;

delete[] AC\_Y\_TREE;

delete[] data;

data = nullptr;

out.close();//关闭文件

}

if (!strcmp(argv[1], "-read")) {

//展示图像PicReader imread;

PicReader imread;

BYTE\* data = nullptr;

UINT x, y;

imread.readPic(argv[2]);

imread.getData(data, x, y);

imread.showPic(data, x, y);

data = nullptr;

delete[] data;

}

cerr << "complete!!!";

return 0;

}