**同济大学计算机系**

**高级程序设计大作业报告**

**IV**

****

**学 号 1950000**

**姓 名 一二三**

**专 业 计算机类**

**班 级 18级计算机二班**

**授课老师 陈宇飞**

**完成日期 2019/7/1**

图像转ASCII小程序大作业

一、实验内容

首先根据OJ给定的要求实现一个Array类，使该类具备有如下功能：①至少能初始化三个维度的数组，即该类至少需要支持三维索引；②具备基本的矩阵操作功能，至少拥有：矩阵加减法，矩阵点乘点除的操作；③支持reshape操作，即从一个一维矩阵reshape成二维或三维矩阵。；④支持获取C风格元数据操作，即从Array中获取一个指针，指针内的数据为你矩阵的元数据。

然后再在该Array类的基础上，藉由OJ给出的Z\_4\_Resources框架，实现一个能够读取图片像素信息，然后将像素内容转化成ASCII字符并以适当的大小显示在cmd窗口的小程序。

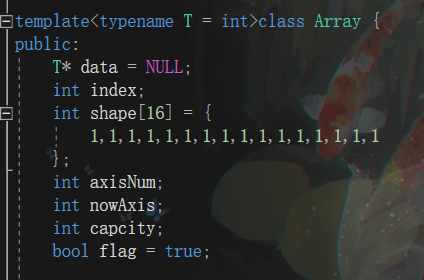
**二、设计思路与功能描述**

**2.1 项目思路说明**

**2.1.1 Array类的实现**

Array类的成员属性基本和OJ提供的成员属性一致，但额外添加了capacity属性，用以存储原数组所占用的空间大小；添加了flag属性，用以存储当前对象在拷贝时是否需要新申请data空间。

其成员属性截图为：



其中data指针用于指向当前对象所包含数据块的首位置，而index值则用于说明当前对象所指向的开始位置，通过两者的搭配使用可以实现对指定维度位数的索引。

shape数组则用于记录该无限维数组每一个维度的容量；axisNum用于记录原数组的维数；nowAxis用于记录当前索引位置的维数。axisNum与nowAxis是否相等是作为析构时是否释放data指针的唯一标准。

在完成对Array类基本属性的说明之后，以下将根据Array类四个功能要求的实现逐一说明Array类的工作原理。

1）支持多维度的索引。

在Array中，可使用的索引方式主要有[]的重载与at成员函数两种，这两种重载都要求能够实现对指定位置数据的增删改查。

为了实现对[]运算符的连续使用，这就要求[]运算符的返回类型依然是Array类，而由于在该过程中还同时能够对原数组的data内容进行修改，所以可知在生成Array返回对象的过程中，应该没有额外申请空间，不同对象的data指针指向的数据块也都是原数组的数据块。

与此同时，每当进行一次[]重载函数计算，就意味着该返回的Array类的当前维度（nowAxis）就会降低一个维度，与此同时，index值也要根据已经[]中的内容对指定维度容量大小的成绩进行增大，由此实现逐步“逼近”用户希望被索引的位置。

该过程的整体逻辑为：Array若目前已是0维，则无法索引，退出程序；若合理则从[]中取出要索引的下一个维度的值，将该值与目前除第一维度容量以外的维度容量相乘，从而得到当前索引指向的index值。之后定义一个返回的Array对象，其data指针与当前Array对象相符，但nowAxis减少了一维，index值进行了细致增加，shape值将当前Array的第一维刨除后进行了重新赋值。再将该对象作为计算结果返回，等待下一步索引或操作。

at成员函数的本质与[]重载相同，但at函数可以同时完成多个维度的索引查找，所以只需要根据at传入参数的个数套上for循环即可。

值得一提的是，由于该过程中生成的Array本质都是在索引计算中的“附带产物”，其data指针所指的数据块都是原始数组的数据块，所以在这些“附带产物”析构的过程，不应该对data空间进行释放——这一过程只需要判断nowAxis与axisNum值是否一样即可。

2）支持矩阵点运算

本功能的实现其实只是单纯的加减符号的多态重载，并不具有很大的难度，但如何让计算结果能够顺利返回到主函数而不在中途被析构掉，才是本题最大的难点所在。

对题目要求进行分析，可知当两个Array进行运算时，产生结果的过程不应该影响到这两个Array的任何属性，所以这就要求运算符重载时返回的新产生对象的data指针不应该和两个Array类公用数据块，而应当是重新申请一个大小为capacity的空间用以存放计算结果。

但新产生的对象本质是重载函数中的局部变量，在返回时势必要调用析构函数，但因为该新对象没有进行索引，nowAxis与axisNum的值一样，所以data中存放的计算结果会被释放掉，导致结果无法传递回主函数。

这是因为重载函数在return时要先调用一次拷贝构造函数，将要return的对象拷贝到一个临时对象中，再藉由该对象作为消息与主函数通信。而在一般情况下，拷贝构造函数就是简单的对要return对象中的内容进行赋值，data指针也会被随之复制过去，这时候data指针与类中定义的局部对象的data指针公用数据块，所以在重载函数结束，析构局部对象时，该数据块被释放，临时对象的data指针也变成了野指针。

为此，只需要在明确后续主函数还需要使用data内容的前提下，将flag值赋为false，然后在拷贝构造函数中，将flag值为false的对象的data新申请一个大小为capacity的空间，然后将原data中的值一一复制过来即可。由此可实现分离两个对象data指针所指数据块的目的。

3）支持reshape操作

由于Array中存储的数据本质上都是一维的，所以所谓的索引、维度都是再以不同的视角来实现对data所指数据块内容的查看。所以只需要将新定义的维度复制到Array中，然后再对axisNum与nowAxis的值加以修改即可。

而在进行索引调用时，假设原数组的容量为[x1],[x2],...,[xn]，当用户调用的索引为[a1],[a2],...[am]（m≤n）时，可得其index的计算式为：



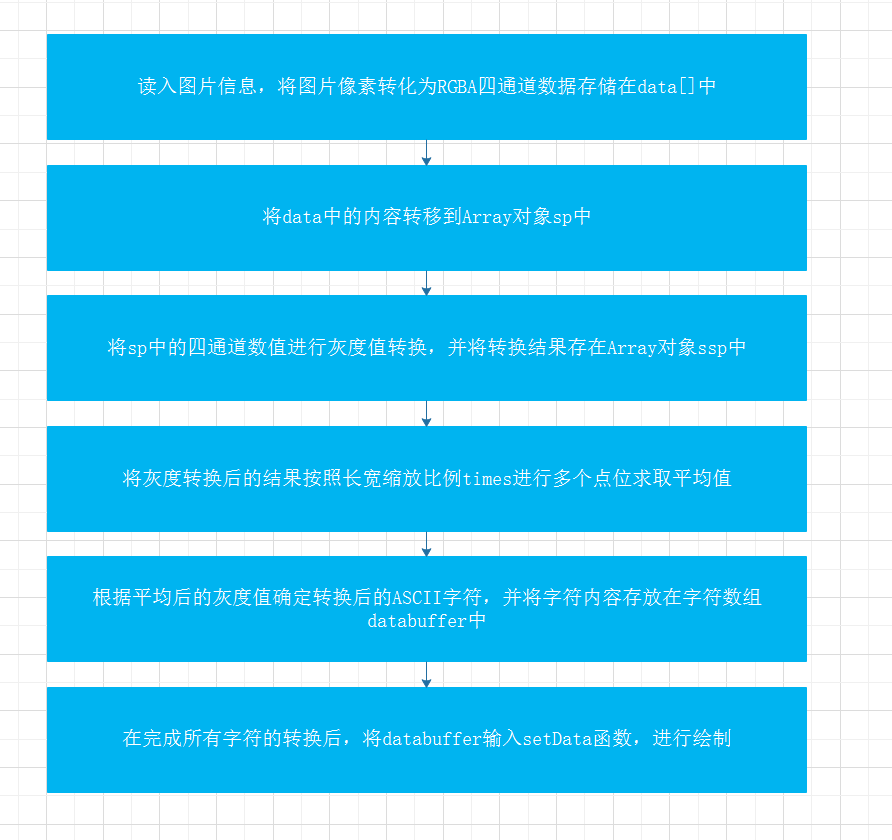
而所谓的reshape，其本质就是对xk的值进行了修改。

4）C风格元素操作

由于Array中的数据都是以线性存储在一个数据块中的，如果用户想通过指针实现类似于C风格的操作时，只需要对data指针返回以供操作即可。

**2.1.2 字符画的实现**

字符画绘制的具体流程图为：



通过流程图不难看出，字符画的绘制本质上就是将要绘制的图片像素进行识别，然后将其转化为对应的字符，具体可分为如下几个步骤：

1）绘制图片的读入

该部分功能已由框架给出，主要是通过winAPI的相关函数，将对应图片的像素信息以R G B A R G B A...的方式存入data[]数组中。

2）data内容转移至sp

为了便于后续操作，这里需要将data中的内容存放入Array对象中，首先定义一个对象sp，其规格为图片长×宽×4（四通道），该对象主要用于将本属于一个图片位置的四个通道数据进行整合，便于后续操作。

3）灰度转换

由于最终的字符画是按照灰度信息进行转换的，所以在这里需要对图片的灰度信息进行计算，具体的计算公式为：



同时，在这一过程中考虑到cmd窗口输出的字符高度为宽度的两倍，所以为了使最终绘制结果不会被“压扁”，所以在这里采用同时读取上下两个相邻位置的灰度值求平均的方式来作为对应位置的灰度值。

4）times倍数压缩

由于受到屏幕大小的限制，在不调整控制台字符大小的前提下，无法完整显示图片，而由于本人使用的为win8系统，无法在C++程序中完成对控制台字符大小的控制，所以只能采用压缩图片分辨率的方式来进行调整。

具体的调整方式为：限制绘制高度为60行、宽度为190个字符，通过将灰度信息长宽等比例缩放至适应大小，得到缩放比times的值（times为一整数）。然后对灰度值Array中记录的所有数据划分为times×times的小格，每个小格内部求取平均值作为该小格元素的灰度值。由此可实现对图片长宽的压缩。

5）转ASCII字符

在本次大作业中，采用的ASCII梯度为OJ上提供的示例梯度，主要包含如下字符：

'M','N','H','Q','$','O','C','?','7','>','!',':','-',';','.'

共计15个字符，在实际转换中，即将原最大值为255的灰度值除以18，并取整，即可得到该灰度对应的ASCII值。

同时，为了方便后续的快速输出，在转换的同时还应该将转换的结果存放在一维数组databuffer中

6）开始绘制

在得到完整的databuffer后，生成一个与databuffer相同大小的一维数组colorbuffer用于存储等数组长度数量的0x0070，用以表示每个字符的位置都应该是“白底黑字”。然后将databuffer与colorbuffer共同送入setData函数，然后通过快速输出对象printer的draw函数，绘制相应的字符画。

**2.1.3 字符视频的实现**

字符视频相较于前两个部分来说较为简单，其主要的实现逻辑为从指定的位置逐一读取视频每一帧的图片信息，在所有图片读取存储结束后开始进行视频的播放。

在播放开始前，首先使用PlaySound函数对视频配套的音频进行播放，然后按照视频的帧率每隔定长时间按顺序输出一次字符画（本实验使用的视频为每秒25帧，所以每40毫米刷新一次）。由此可实现字符视频的播放。

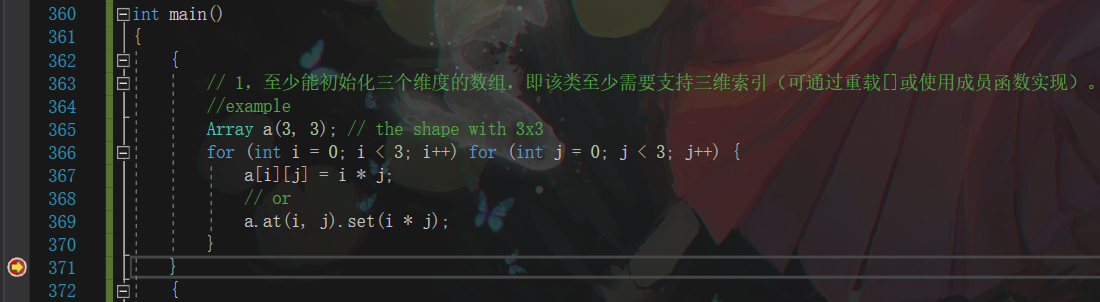
**2.2 功能描述**

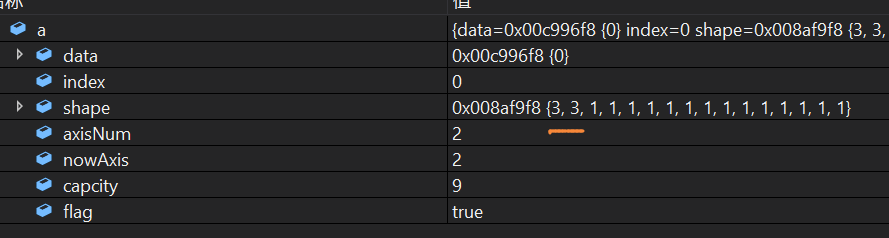
**2.2.1 Array类说明**

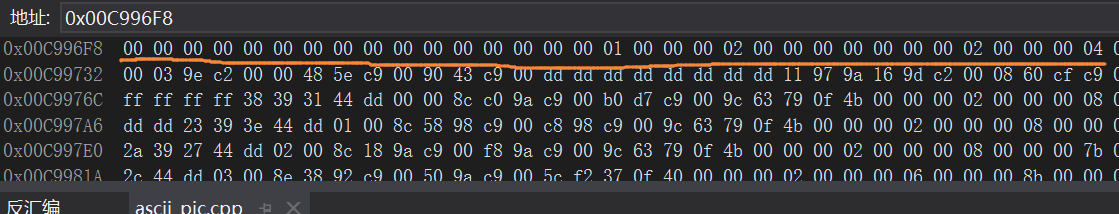
根据OJ上提供的测试用main函数，可以使用VS的内存查看器对Array中四个功能的实现进行确认，测试过程截图如下：

1）功能①检验

在main函数中功能1结束时添加断点，查看a对象的内部情况，可得：





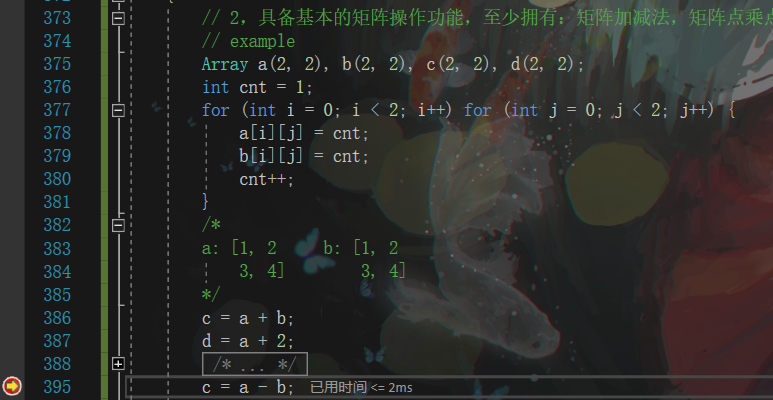


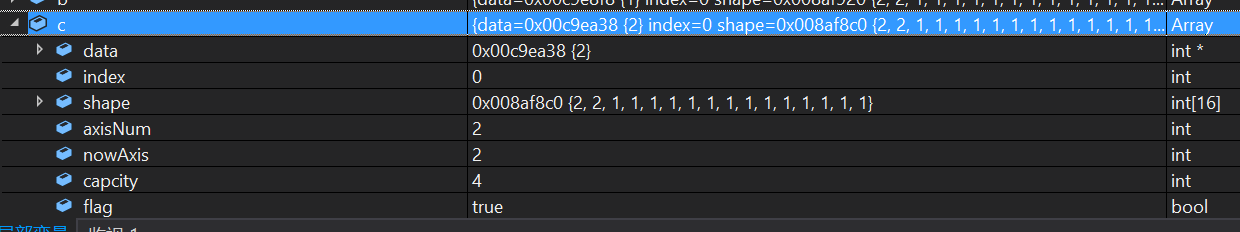
根据断点内容不难看出，此时a的shape确为3×3，而且通过查看地址内存，发现data中的数据确为0，0，0，0，1，2，0，2，4。

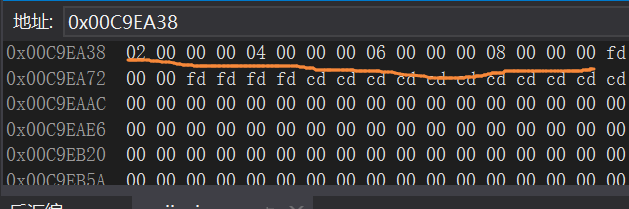
故可认为Array的功能①能够实现。

2）功能②检验

在main函数中在功能2加法运算结束时添加断点，查看c对象的内部情况，可得：





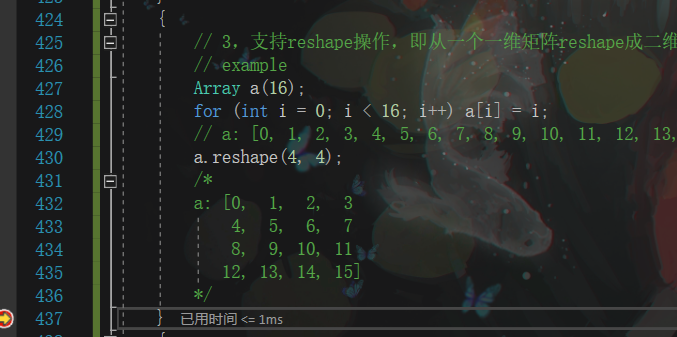


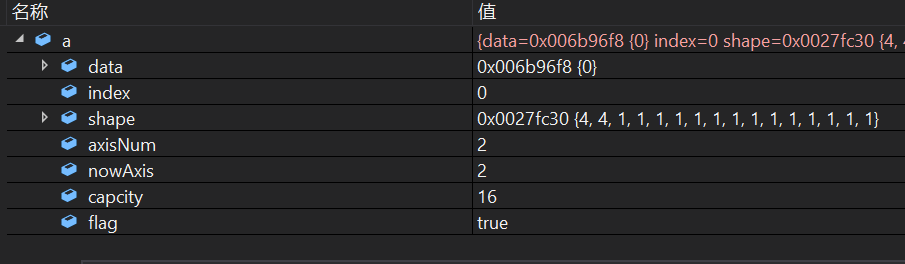
通过上述内容可以看出，c的计算结果确为2，4，6，8，而且shape的形状也与预期相同，故可认为加法运算正确。考虑到剩余几个四则运算的本质与加法相同，故不在此进行赘述。

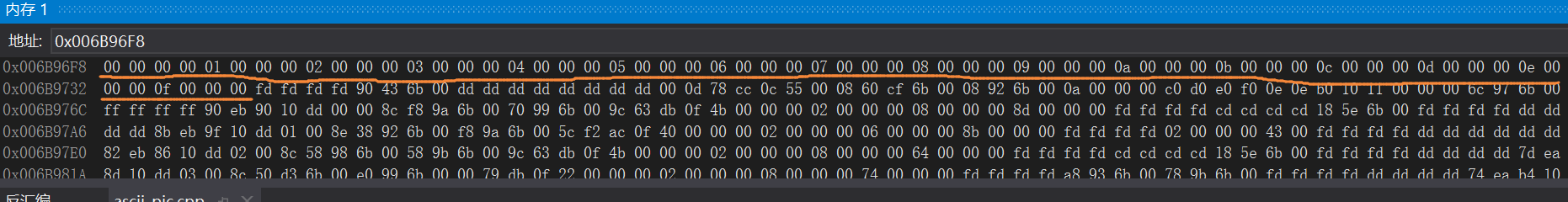
综上所述，可认为Array的功能②能够实现。

3）功能③检验

在main函数功能3结束时添加断点，查看a对象的内部情况，可得：







通过上述内容可以看出，a的计算结果的确是对shape的形状进行了重构，与此同时，data数据块中存储的内容并没有发生变化。

故可认为Array的功能③能够实现。

4）功能④检验

运行至main函数结束，查看cmd窗口的输出，可得：



故可认为在功能4中，确能够将data的指针反馈给int\* b，b也能够通过C风格的方式实现数据的访问。

故可认为Array的功能④能够实现。

**2.2.2 提交程序说明**

在运行本次大作业的程序时，首先可以进入菜单选择界面，其中a)~k)均为ASCII字符画的输出，l)表示加分项视频的播放，q)表示退出。

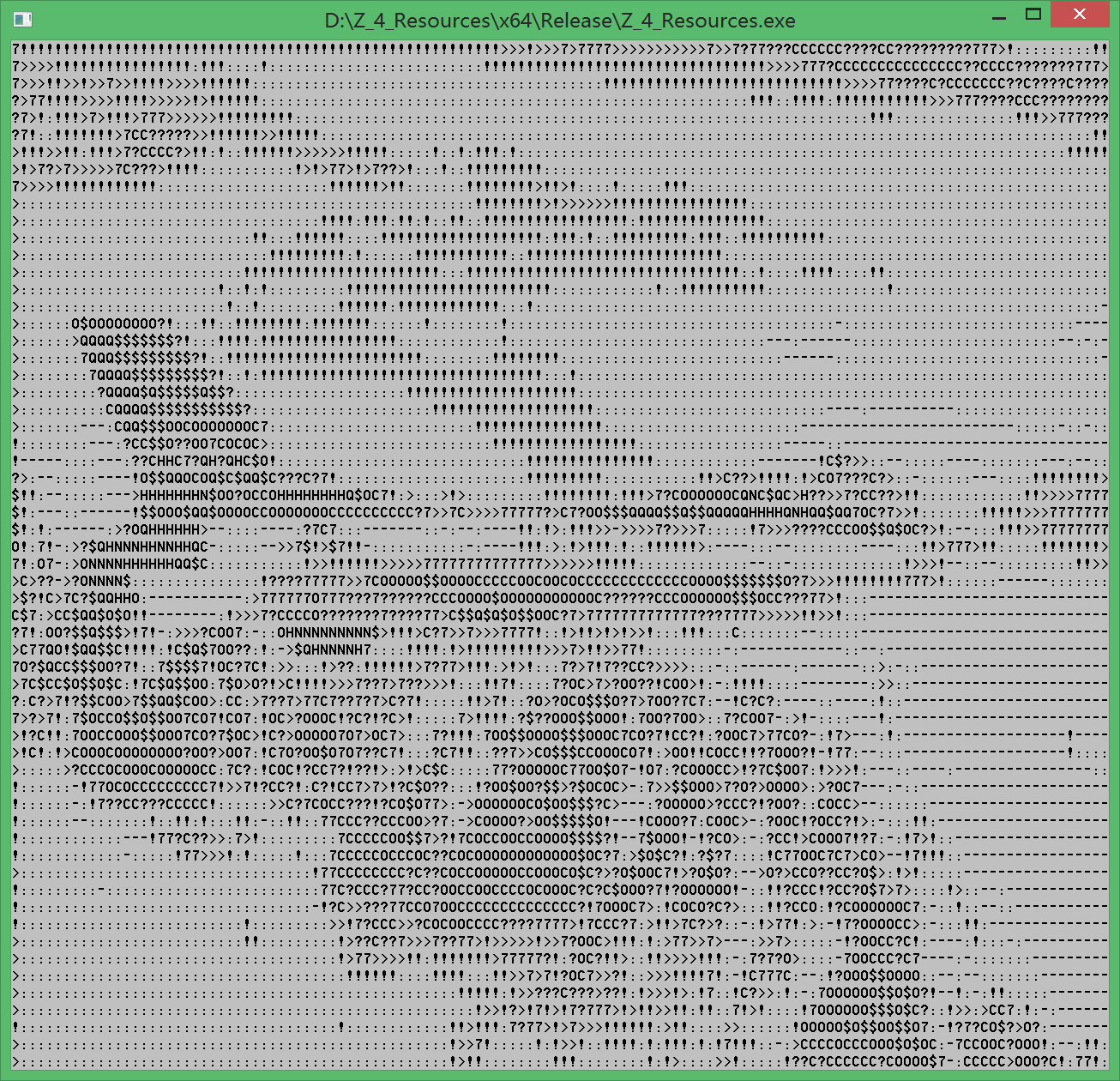
每当用户完成字符画的查看后，按下回车键即可再次返回本菜单界面直至输入命令'q'，退出等待，程序运行结束。

该菜单的界面仿照了上学期第一次大作业的菜单界面，界面截图为：



以下将按照菜单的顺序依次对相关功能进行展示：

1）airplane.jpg



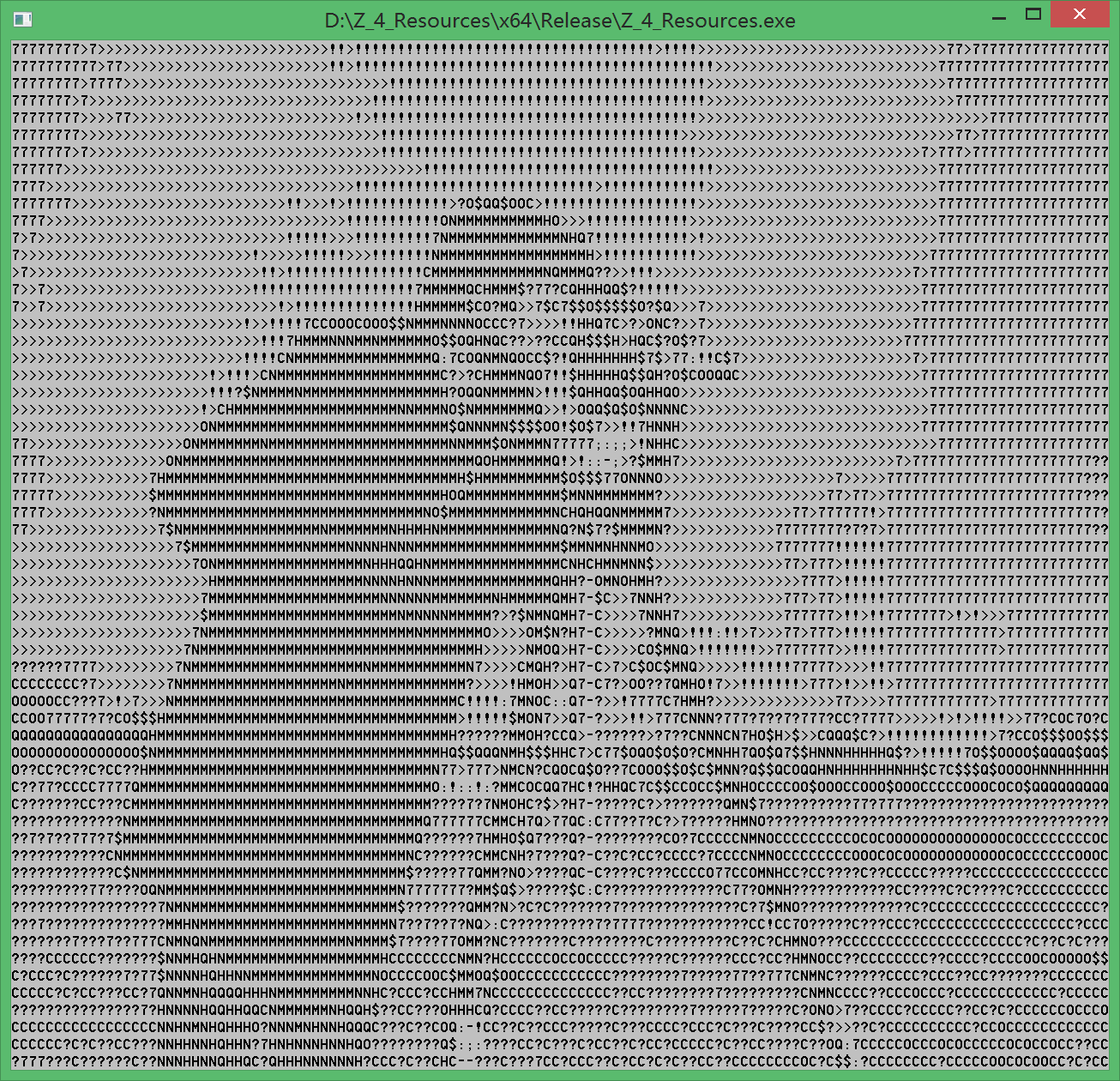
2）baboon.jpg



3）barbara.jpg



4）cameraman.jpg



5）compa.png



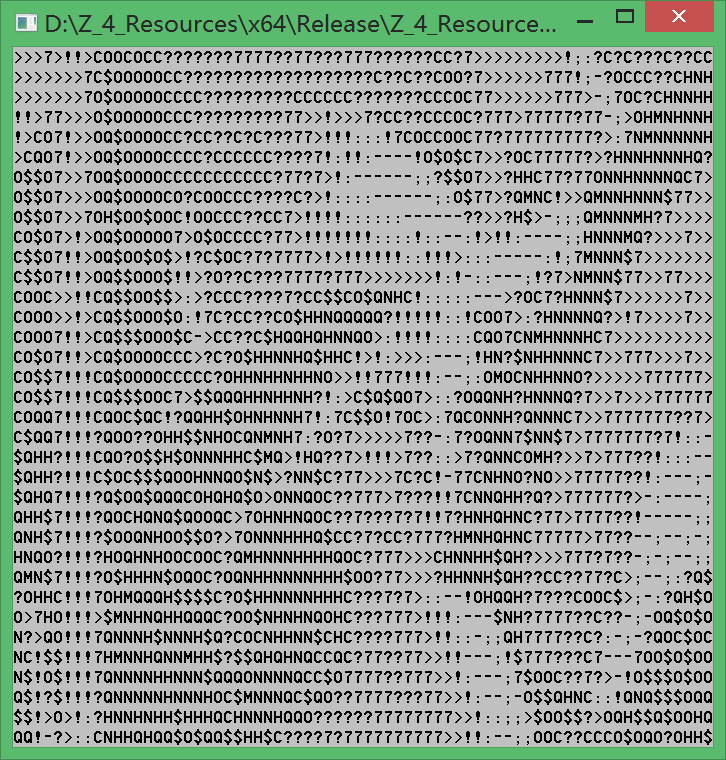
6）goldhill.jpg



7）lena.jpg



8）lena1.jpg



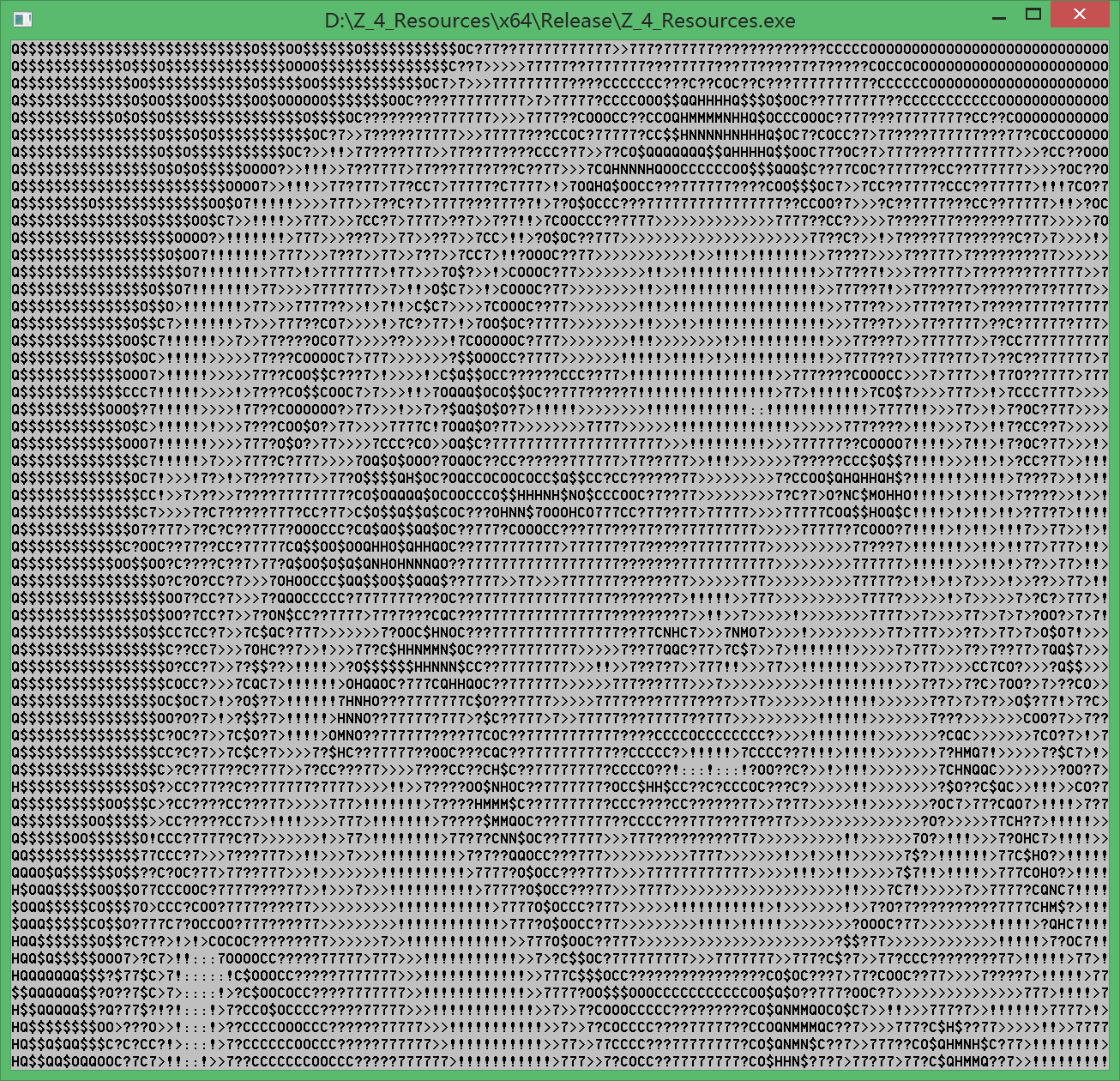
9）milkdrop.jpg



10）peppers/jpg



11）woman.jpg

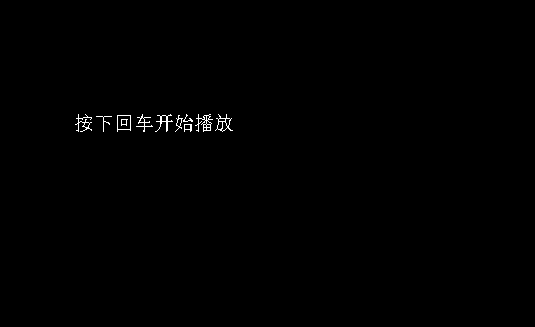


12）sea.mp4

这里我选用的影片是毛不易与b站合作在今年5月20日发布于B站的毕业季MV《入海》的第一个高潮部分片段。

当我们选择l功能后程序开始加载，加载结束后会提示可以开始播放。由于图片数量多达1011张，所以建议在正式使用时采用Release模式，该模式下加载40秒左右即可加载完毕。

提示语截图为：



在此页面按下回车后，可以开始播放视频，同时音乐响起，以下为播放过程中的截图：



三、项目亮点与情况说明

**3.1 项目亮点**

①本次大作业整体是在OJ所给的框架下完成的，自认为在Array类的析构、拷贝构造函数等操作的处理还是具备较高的鲁棒性的。

②本次大作业完成了OJ的所有内容，以及加分项的内容实现，且没有使用任何外部类。

③本次大作业仿照了类似于上学期第一次大作业的菜单设计，能够在一次运行中多次查看，提高了一定的便捷性。

④在本次大作业的编写要求之外，与雷丁瑞同学使用vector的思想还额外进行了一版专门用于分享的字符画程序。

**3.2 诸多情况说明**

①自身windows系统较低，限制了最终成果美观性。

由于本人的开发环境为win8系统，查阅了很多资料也没有找到切实可用的在c++程序中更改控制台字符的函数，只能通过修改本地的控制台配置实现。

但考虑到助教在核验时无法第一时间知道我对控制台进行了额外设置，可能会因为超出屏幕范围而导致助教对产生的字符画不明所以，所以我最终还是决定采用在times倍率压缩图片的方式来保证图片在一个合理的范围内显示。

②在框架外没有进行任何形式的拓展

首先是没有使用先前vector思想，因为vector最大的特点是能够无限制地像其中push元素，与本次大作业给出框架所需的Array类特点不符，所以在正是提交的版本中，没有使用vector的相关元素。

其次是没有做将字符画进一步渲染为彩色的工作。这主要是因为我本人是大二转专业的学生，同时还提前修读了大三的软件工程课程与Web技术课程，这导致我整个6月都在忙于各种各样的大作业，所以并没有能够分配很多时间到最后一次大作业，致使最终没有进行进一步的改进。

四、遇到的问题与解决方法

**4.1 遇到的问题**

在刚开始尝试视频播放的过程中，出现了音画不同步的错误。例如在《入海》MV中，歌声响起与镜头从出租车切到下一场景是同时发生的，但在实际测试的过程中，发现总是会出现图片播放或快或慢的情况。

**4.2 解决方法**

这个问题的解决其实耗费了我很长时间，刚开始的时候我发现的问题是视频帧率的问题，就是我的程序是按照每秒24帧进行编写的，但是后来我在查看原视频属性时，发现我找的视频是每秒25帧的，这里算是发现的第一个问题。

其次是我对于运行每隔40ms的时间控制，刚开始的时候我是直接让程序Sleep(40)，但是发现画面播放地相当缓慢，都快0.5倍速了，后来我意识到程序在绘制字符画的过程中也会消耗不可忽略的时间，所以直接Sleep(40)无异于极大的拖慢了字符画面的更新时间。

于是我通过VS的调试程序，测算了在Release模式下，每绘制一遍页面需要大约11ms左右的时间，所以我将程序暂停时间更改为了Sleep(29)。这时候再运行程序，发现就能够很好地音画同步显示了，我自以为已经万事大吉了。

但后来的某一天，在我复习计算机组成原理内容的时候，我突然想到，绘制页面的速度应该与电脑的性能、当前进程的占用状态等都有关系，所以我测出的11ms应该只是我的电脑在当时的进程占用状态下执行相应功能所需要的时间。

于是我让同样的程序在Pr渲染视频的同时执行，果不其然，音画不同步的现象又出现了，但我总不能让用户在每次播放前测试一遍。

后来我受到了本学期第一个“类大作业”的clock时钟的启发，我当时为了避免出现抢秒、跳秒的状况，采用了不停获取当前秒数，当当前秒数与上一个获取的秒数不一样时，重新渲染一次时钟的方式。

于是，我将程序改为了调整播放，即所有时间都与开始时间相比较，每当当前时间与开始时间的差值是40的倍数时，说明此时应该播放新的一页，则进入页面的绘制工作。

经过上述改动之后，发现永久解决了音画不同步的问题，我在多个CPU不同占用情况的条件下进行测试，发现播放均正确。

五、心得体会

**5.1 额外的发现**

在我完成本次大作业的过程中，出于好奇，对wav格式进行了一些小范围的搜索，主要是想了解为什么PlaySound只能播放.wav格式的音乐。

后来随着我查询的深入，我慢慢了解到wav其实是微软公司专门为windows开发的一种声音形式，在多种windows插件中都有这较为广泛的应用；同时待遇相同的还有wmv等。

面对这样一种大家公认“体积太大”的音频格式，微软凭借其强大的流水线与广阔的软件服务，硬生生将这个格式带入了主流音频格式——感觉颇有一番腾讯企鹅靠QQ社交降维打击其他领域的架势。

这不禁让我深深地感受到产业化的重要性，只要当一个企业或国家能够在一个领域拥有一个庞大的产业链，它就可以凭借着“多数人都在用”这个唯一的理由把握相当大的话语权，最终凭借着自己制定的行业标准进一步将自己利益滚雪球。

目前我国处于领先地位的只有特高压输电等几个小小的领域，而欧美在计算机、芯片、通讯等等方面对我国的产业壁垒也让人望洋兴叹。

不得不说，我国确实与欧美还有相当大的差距，“赶英超美”的重担也确实需要我们一代人一代人去努力实现。

**5.2 写在最后**



六、源代码

1.Picreader.h

1. */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
2. \* ！注意！                                                         \*
3. \* 本头文件中为你封装了WinAPI中WIC底层函数，方便你进行图片读取而不必引  \*
4. \* 入或安装其他的外部库，但是我们有一定的约束条件，请你仔细阅读以下规定  \*
5. \*     本头文件中任何没有TO-DO的地方请你不要修改，若函数存在问题，      \*
6. \* 请及时联系老师或助教！                                            \*
7. \*     每一个TO-DO块以TO-DO：说明 (TO-DO) END 结束，具体可看下方代码  \*
8. \*     readPic()函数为你封装了WinAPI中的方法，可以将图片读取为RGBA的   \*
9. \* bitmap数据，但这并不代表你可以通过修改这个函数直接达到读取灰度图的   \*
10. \* 目的。                                                           \*
11. \*     getData()是你最终需要完善的函数，将读取出来的一维BYTE数组转换   \*
12. \* 成你实现的Array类。                                              \*
13. \*     testReader()是demo中提供读取数据的其中一个思路。               \*
14. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
15. #ifndef PIC\_READER\_H
16. #define PIC\_READER\_H
17. #include <windows.h>
18. #include <wincodec.h>
19. #include <commdlg.h>
20. #include <iostream>
21. template<typename T = int>class Array {
22. public:
23. T\* data = NULL;
24. int index;
25. int shape[16] = {
26. 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
27. };
28. int axisNum;
29. int nowAxis;
30. int capcity;
31. bool flag = true;
32. Array() {}
33. template<typename ...Args>
34. Array(Args... args) {
35. *// 获取参数包大小并转换为size\_t数组*
36. auto num = sizeof...(args);
37. size\_t list[] = { args... };
38. int size = 1;
39. axisNum = num;
40. for (int i = 0; i < axisNum; i++) {
41. shape[i] = list[i];
42. size \*= shape[i];
43. }
44. capcity = size;
45. data = new T[size];
46. index = 0;
47. nowAxis = axisNum;
48. }
49. template<typename ...Args>
50. Array at(Args... args) {
51. *// 获取参数包大小并转换为size\_t数组*
52. auto num = sizeof...(args);
53. size\_t list[] = { args... };
54. if (num > this->nowAxis) {
55. exit(-1);
56. }
57. int size = 0, size1 = 1;
58. for (int i = 0; i < num; i++) {
59. size1 = list[i];
60. for (int j = i + 1; j < this->nowAxis; j++) {
61. size1 \*= shape[i];
62. }
63. size += size1;
64. }
65. Array temp;
66. temp.axisNum = this->axisNum;
67. temp.data = this->data;
68. temp.index = this->index + size;
69. temp.nowAxis = this->nowAxis - num;
70. temp.capcity = this->capcity;
71. int uhi = 0;
72. for (int i = 0; i < this->nowAxis; i++) {
73. if (i < num) {
74. continue;
75. }
76. temp.shape[uhi] = this->shape[i];
77. uhi++;
78. }
79. return temp;
80. }
81. template<typename ...Args>
82. void reshape(Args... args) {
83. *// 获取参数包大小并转换为size\_t数组*
84. auto num = sizeof...(args);
85. size\_t list[] = { args... };
86. int uhi = 1, uhj = 1;
87. for (int i = 0; i < num; i++) {
88. uhi \*= list[i];
89. uhj \*= this->shape[i];
90. }
91. if (uhi != uhj) {
92. exit(-1);
93. }
94. for (int i = 0; i < 16; i++) {
95. if (i < num) {
96. shape[i] = list[i];
97. }
98. else {
99. shape[i] = 1;
100. }
101. }
102. axisNum = num;
103. nowAxis = axisNum;
104. this->index = 0;
105. }
106. T\* get\_content() {
107. return data;
108. }
109. void set(int value) {
110. this->data[this->index] = value;
111. }
112. Array operator[](int index) {
113. *// 在这里修改子矩阵的nowAxis的值以及其他有必要的值，以返回一个子矩阵*
114. if (this->nowAxis == 0) {
115. exit(-1);
116. }
117. int size = index;
118. for (int i = 1; i < this->nowAxis; i++) {
119. size \*= this->shape[i];
120. }
121. Array temp;
122. temp.axisNum = this->axisNum;
123. temp.data = this->data;
124. temp.index = this->index + size;
125. temp.nowAxis = this->nowAxis - 1;
126. temp.capcity = this->capcity;
127. for (int i = 1; i < this->nowAxis; i++) {
128. temp.shape[i - 1] = this->shape[i];
129. }
130. return temp;
131. }
132. ~Array() {
133. if (axisNum == nowAxis) {
134. delete[] data;
135. }
136. }
137. Array& operator=(Array sp) {
138. this->axisNum = sp.axisNum;
139. this->capcity = sp.capcity;
140. this->index = sp.index;
141. this->nowAxis = sp.nowAxis;
142. for (int i = 0; i < this->axisNum; i++) {
143. this->shape[i] = sp.shape[i];
144. }
145. this->data = new T[this->capcity];
146. for (int i = 0; i < this->capcity; i++) {
147. this->data[i] = sp.data[i];
148. }
149. this->flag = true;
150. return \*this;
151. }
152. Array(const Array& sp) {
153. this->axisNum = sp.axisNum;
154. this->capcity = sp.capcity;
155. this->index = sp.index;
156. this->nowAxis = sp.nowAxis;
157. for (int i = 0; i < this->axisNum; i++) {
158. this->shape[i] = sp.shape[i];
159. }
160. if (sp.flag) {
161. this->data = sp.data;
162. }
163. else {
164. this->data = new T[this->capcity];
165. for (int i = 0; i < this->capcity; i++) {
166. this->data[i] = sp.data[i];
167. }
168. this->flag = true;
169. }
170. }
171. Array operator=(T data) {
172. this->data[index] = data;
173. return \*this;
174. }
175. operator int() {
176. return data[index];
177. }
178. Array operator+(Array &sp) {
179. if (this->capcity != sp.capcity) {
180. exit(-1);
181. }
182. Array temp;
183. temp.axisNum = this->axisNum;
184. temp.capcity = this->capcity;
185. temp.index = this->index;
186. temp.nowAxis = this->nowAxis;
187. temp.data = new T[this->capcity];
188. temp.flag = false;
189. for (int i = 0; i < this->axisNum; i++) {
190. temp.shape[i] = this->shape[i];
191. }
192. for (int i = 0; i < this->capcity; i++) {
193. temp.data[i] = this->data[i] + sp.data[i];
194. }
195. return temp;
196. }
197. Array operator+(int sp) {
198. Array temp;
199. temp.axisNum = this->axisNum;
200. temp.capcity = this->capcity;
201. temp.index = this->index;
202. temp.nowAxis = this->nowAxis;
203. temp.data = new T[this->capcity];
204. temp.flag = false;
205. for (int i = 0; i < this->axisNum; i++) {
206. temp.shape[i] = this->shape[i];
207. }
208. for (int i = 0; i < this->capcity; i++) {
209. temp.data[i] = this->data[i] + sp;
210. }
211. return temp;
212. }
213. Array operator-(Array &sp) {
214. if (this->capcity != sp.capcity) {
215. exit(-1);
216. }
217. Array temp;
218. temp.axisNum = this->axisNum;
219. temp.capcity = this->capcity;
220. temp.index = this->index;
221. temp.nowAxis = this->nowAxis;
222. temp.data = new T[this->capcity];
223. temp.flag = false;
224. for (int i = 0; i < this->axisNum; i++) {
225. temp.shape[i] = this->shape[i];
226. }
227. for (int i = 0; i < this->capcity; i++) {
228. temp.data[i] = this->data[i] - sp.data[i];
229. }
230. return temp;
231. }
232. Array operator-(int sp) {
233. Array temp;
234. temp.axisNum = this->axisNum;
235. temp.capcity = this->capcity;
236. temp.index = this->index;
237. temp.nowAxis = this->nowAxis;
238. temp.data = new T[this->capcity];
239. temp.flag = false;
240. for (int i = 0; i < this->axisNum; i++) {
241. temp.shape[i] = this->shape[i];
242. }
243. for (int i = 0; i < this->capcity; i++) {
244. temp.data[i] = this->data[i] - sp;
245. }
246. return temp;
247. }
248. Array operator\*(Array &sp) {
249. if (this->capcity != sp.capcity) {
250. exit(-1);
251. }
252. Array temp;
253. temp.axisNum = this->axisNum;
254. temp.capcity = this->capcity;
255. temp.index = this->index;
256. temp.nowAxis = this->nowAxis;
257. temp.data = new T[this->capcity];
258. temp.flag = false;
259. for (int i = 0; i < this->axisNum; i++) {
260. temp.shape[i] = this->shape[i];
261. }
262. for (int i = 0; i < this->capcity; i++) {
263. temp.data[i] = this->data[i] \* sp.data[i];
264. }
265. return temp;
266. }
267. Array operator\*(int sp) {
268. Array temp;
269. temp.axisNum = this->axisNum;
270. temp.capcity = this->capcity;
271. temp.index = this->index;
272. temp.nowAxis = this->nowAxis;
273. temp.data = new T[this->capcity];
274. temp.flag = false;
275. for (int i = 0; i < this->axisNum; i++) {
276. temp.shape[i] = this->shape[i];
277. }
278. for (int i = 0; i < this->capcity; i++) {
279. temp.data[i] = this->data[i] \* sp;
280. }
281. return temp;
282. }
283. Array operator/(Array &sp) {
284. if (this->capcity != sp.capcity) {
285. exit(-1);
286. }
287. Array temp;
288. temp.axisNum = this->axisNum;
289. temp.capcity = this->capcity;
290. temp.index = this->index;
291. temp.nowAxis = this->nowAxis;
292. temp.data = new T[this->capcity];
293. temp.flag = false;
294. for (int i = 0; i < this->axisNum; i++) {
295. temp.shape[i] = this->shape[i];
296. }
297. for (int i = 0; i < this->capcity; i++) {
298. temp.data[i] = this->data[i] / sp.data[i];
299. }
300. return temp;
301. }
302. Array operator/(int sp) {
303. Array temp;
304. temp.axisNum = this->axisNum;
305. temp.capcity = this->capcity;
306. temp.index = this->index;
307. temp.nowAxis = this->nowAxis;
308. temp.data = new T[this->capcity];
309. temp.flag = false;
310. for (int i = 0; i < this->axisNum; i++) {
311. temp.shape[i] = this->shape[i];
312. }
313. for (int i = 0; i < this->capcity; i++) {
314. temp.data[i] = this->data[i] / sp;
315. }
316. return temp;
317. }
318. };
319. template <typename T>
320. inline void SafeRelease(T \*&p) {
321. if (nullptr != p) {
322. p->Release();
323. p = nullptr;
324. }
325. }
326. class PicReader {
327. public:
328. PicReader();
329. ~PicReader();
330. void readPic(LPCSTR);
331. void getData(Array<int> &spp, UINT &xx, UINT &yy);
332. void testReader(BYTE \*&,UINT &, UINT &);
333. private:
334. void init();
335. bool checkHR(HRESULT);
336. void quitWithError(LPCSTR);
337. HWND                    hWnd;
338. HANDLE                  hFile;
339. IWICImagingFactory     \*m\_pIWICFactory;
340. IWICFormatConverter    \*m\_pConvertedSourceBitmap;
341. */\*TO-DO：这里可能会增加你需要的内部成员 END\*/*
342. };
343. PicReader::PicReader() : m\_pConvertedSourceBitmap(nullptr), m\_pIWICFactory(nullptr) {
344. init();
345. }
346. PicReader::~PicReader() {
347. if (hFile != NULL) CloseHandle(hFile);
348. SafeRelease(m\_pConvertedSourceBitmap);
349. SafeRelease(m\_pIWICFactory);
350. CoUninitialize();
351. }
352. bool PicReader::checkHR(HRESULT hr) {
353. return (hr < 0);
354. }
355. void PicReader::quitWithError(LPCSTR message) {
356. MessageBoxA(hWnd, message, "Application Error", MB\_ICONEXCLAMATION | MB\_OK);
357. quick\_exit(0xffffffff);
358. }
359. void PicReader::init() {
360. hWnd = GetForegroundWindow();
361. *// Enables the terminate-on-corruption feature.*
362. HeapSetInformation(nullptr, HeapEnableTerminationOnCorruption, nullptr, 0);
363. HRESULT hr = S\_OK;
364. *//Init the WIC*
365. hr = CoInitializeEx(nullptr, COINIT\_APARTMENTTHREADED | COINIT\_DISABLE\_OLE1DDE);
366. *// Create WIC factory*
367. hr = CoCreateInstance(
368. CLSID\_WICImagingFactory,
369. nullptr,
370. CLSCTX\_INPROC\_SERVER,
371. IID\_PPV\_ARGS(&m\_pIWICFactory)
372. );
373. *// Throw error if create factor failed*
374. if (checkHR(hr)) { quitWithError("Init Reader Failed"); }
375. }
376. void PicReader::readPic(LPCSTR fileName) {
377. HRESULT hr = S\_OK;
378. *// Create a File Handle (WinAPI method not std c)*
379. if (hFile != NULL) CloseHandle(hFile);
380. hFile = CreateFileA(fileName, GENERIC\_READ, FILE\_SHARE\_READ, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);
381. if (GetLastError() == ERROR\_FILE\_NOT\_FOUND) {
382. quitWithError("Cannot find such file, please retry or check the access");
383. }
384. *// Create a decoder*
385. IWICBitmapDecoder \*pDecoder = nullptr;
386. hr = m\_pIWICFactory->CreateDecoderFromFileHandle((ULONG\_PTR)hFile, nullptr, WICDecodeMetadataCacheOnDemand, &pDecoder);
387. if (checkHR(hr)) { quitWithError("Create Decoder Failed"); }
388. *// Retrieve the first frame of the image from the decoder*
389. IWICBitmapFrameDecode \*pFrame = nullptr;
390. hr = pDecoder->GetFrame(0, &pFrame);
391. if (checkHR(hr)) { quitWithError("Get Frame Failed"); }
392. *// Format convert the frame to 32bppRGBA*
393. SafeRelease(m\_pConvertedSourceBitmap);
394. hr = m\_pIWICFactory->CreateFormatConverter(&m\_pConvertedSourceBitmap);
395. if (checkHR(hr)) { quitWithError("Get Format Converter Failed"); }
396. hr = m\_pConvertedSourceBitmap->Initialize(pFrame, GUID\_WICPixelFormat32bppRGBA, WICBitmapDitherTypeNone, nullptr, 0.f, WICBitmapPaletteTypeCustom);
397. if (checkHR(hr)) { quitWithError("Init Bitmap Failed"); }
398. *// Clean memory*
399. SafeRelease(pDecoder);
400. SafeRelease(pFrame);
401. }
402. void PicReader::getData(Array<int> &spp, UINT &xx, UINT &yy) {
403. HRESULT hr = S\_OK;
404. *// Get the size of Image*
405. UINT x, y;
406. hr = m\_pConvertedSourceBitmap->GetSize(&x, &y);
407. if (checkHR(hr)) { quitWithError("Check Bitmap Size Failed"); }
408. *// Create the buffer of pixels, the type of BYTE is unsigned char*
409. BYTE \*data;
410. data = new BYTE[x \* y \* 4];
411. memset(data, 0, x \* y \* 4);
412. *// Copy the pixels to the buffer*
413. UINT stride = x \* 4;
414. hr = m\_pConvertedSourceBitmap->CopyPixels(nullptr, stride, x \* y \* 4, data);
415. if (checkHR(hr)) { quitWithError("Copy Pixels Failed"); }
416. */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
417. \*  TO-DO:                                                         \*
418. \*                                                                 \*
419. \*  实现一个Array类，并将上面的data转存至你的Array内                  \*
420. \*                                                                 \*
421. \*  数据说明：从Bitmap Copy出来的数据，每4个为一组代表一个像素         \*
422. \*           数据为一个长度为图像的(长\*宽\*4)的一维数组                \*
423. \*           即数据排布为 R G B A R G B A R G B A.....              \*
424. \*                                                                 \*
425. \*  ！注意！  你仅可以只改动从此开始到下一个TO-DO END位置的代码！       \*
426. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
427. Array<BYTE> sp(x, y, 4);
428. int hh = 0;
429. for (int i = 0; i < y; i++) {
430. for (int j = 0; j < x; j++) {
431. for (int k = 0; k < 4; k++) {
432. sp[j][i][k] = data[hh];
433. hh++;
434. }
435. }
436. }
437. spp.axisNum = 2;
438. spp.capcity = y / 2 \* x;
439. spp.index = 0;
440. spp.nowAxis = 2;
441. spp.shape[0] = x;
442. spp.shape[1] = y / 2;
443. spp.data = new int[spp.capcity];
444. int t1, t2, t3;
445. int G1, G2;
446. for (int i = 0; i < x; i++) {
447. for (int j = 0; j < y / 2; j++) {
448. t1 = sp[i][2 \* j][0];
449. t2 = sp[i][2 \* j][1];
450. t3 = sp[i][2 \* j][2];
451. G1 = (t1 \* 299 + t2 \* 587 + t3 \* 114 + 500) / 1000;
452. t1 = sp[i][2 \* j + 1][0];
453. t2 = sp[i][2 \* j + 1][1];
454. t3 = sp[i][2 \* j + 1][2];
455. G2 = (t1 \* 299 + t2 \* 587 + t3 \* 114 + 500) / 1000;
456. spp[i][j] = (G1 + G2 + 1) / 2;
457. }
458. }
459. xx = x;
460. yy = y / 2;
461. delete[] data;
462. */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
463. \*  TO-DO END                                                      \*
464. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
465. *// Close the file handle*
466. CloseHandle(hFile);
467. hFile = NULL;
468. }
469. void PicReader::testReader(BYTE\* &\_out, UINT& \_x, UINT& \_y){
470. HRESULT hr = S\_OK;
471. *// Get the size of Image*
472. UINT x, y;
473. hr = m\_pConvertedSourceBitmap->GetSize(&x, &y);
474. if (checkHR(hr)) { quitWithError("Check Bitmap Size Failed"); }
475. *// Create the buffer of pixels, the type of BYTE is unsigned char*
476. BYTE \*data;
477. data = new BYTE[x \* y \* 4];
478. memset(data, 0, x \* y \* 4);
479. *// Copy the pixels to the buffer*
480. UINT stride = x \* 4;
481. hr = m\_pConvertedSourceBitmap->CopyPixels(nullptr, stride, x \* y \* 4, data);
482. if (checkHR(hr)) { quitWithError("Copy Pixels Failed"); }
483. \_out = data; \_x = x; \_y = y;
484. *// Close the file handle*
485. CloseHandle(hFile);
486. hFile = NULL;
487. }
488. #endif

2.FastPrinter

1. */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
2. \* ！注意！                                                         \*
3. \* 本头文件中为你封装了WinAPI中有关Console绘制的底层函数，可以帮助你快  \*
4. \* 速绘制你想要的输出，效率比printf+cls高出很多。                     \*
5. \* 函数使用详见demo.cpp中的几个示例。                                \*
6. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
7. #ifndef FAST\_PRINTER\_H
8. #define FAST\_PRINTER\_H
9. #include <windows.h>
10. */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
11. \*  TO-DO:                                                         \*
12. \*                                                                 \*
13. \*  本文件你可以自由进行修改，如将其中的一些接收参数设置为你实现的Array  \*
14. \*  或为了配合你的实现进行一些便携化成员函数的编写等，甚至自己重新实现   \*
15. \*  一个更高效的。                                                  \*
16. \*                                                                 \*
17. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
18. namespace fp\_color {
19. *// f is the foreground color b is the background color*
20. *// console color format: (f | b)*
21. const SHORT f\_black = 0;
22. const SHORT f\_blue = 0x0001;
23. const SHORT f\_green = 0x0002;
24. const SHORT f\_aqua = 0x0003;
25. const SHORT f\_red = 0x0004;
26. const SHORT f\_purple = 0x0005;
27. const SHORT f\_yellow = 0x0006;
28. const SHORT f\_white = 0x0007;
29. const SHORT f\_gray = 0x0008;
30. const SHORT f\_l\_blue = 0x0009;
31. const SHORT f\_l\_green = 0x000A;
32. const SHORT f\_l\_aqua = 0x000B;
33. const SHORT f\_l\_red = 0x000C;
34. const SHORT f\_l\_purple = 0x000D;
35. const SHORT f\_l\_yellow = 0x000E;
36. const SHORT f\_l\_white = 0x000F;
37. const SHORT b\_black = 0;
38. const SHORT b\_blue = 0x0010;
39. const SHORT b\_green = 0x0020;
40. const SHORT b\_aqua = 0x0030;
41. const SHORT b\_red = 0x0040;
42. const SHORT b\_purple = 0x0050;
43. const SHORT b\_yellow = 0x0060;
44. const SHORT b\_white = 0x0070;
45. const SHORT b\_gray = 0x0080;
46. const SHORT b\_l\_blue = 0x0090;
47. const SHORT b\_l\_green = 0x00A0;
48. const SHORT b\_l\_aqua = 0x00B0;
49. const SHORT b\_l\_red = 0x00C0;
50. const SHORT b\_l\_purple = 0x00D0;
51. const SHORT b\_l\_yellow = 0x00E0;
52. const SHORT b\_l\_white = 0x00F0;
53. }
54. class FastPrinter {
55. public:
56. FastPrinter(DWORD, DWORD);
57. FastPrinter(DWORD, DWORD, WORD);
58. ~FastPrinter();
59. void setData(const char\*, const WORD\*);
60. void setData(const char\*, const WORD\*, SMALL\_RECT);
61. void setRect(SMALL\_RECT, const char, const WORD);
62. void fillRect(SMALL\_RECT, const char, const WORD);
63. void setText(COORD, const char\*, const WORD, const WORD);
64. void setText(COORD, const char\*, const WORD);
65. void setText(COORD, const char\*);
66. void cleanSrceen();
67. void draw(bool);
68. private:
69. HANDLE hOutput, hOutBuf, hTmpBuf;
70. COORD coordBufSize;
71. COORD coordBufCoord;
72. DWORD bytes = 0;
73. DWORD sizeX, sizeY;
74. char\* dataGrid;
75. WORD\* colorGrid;
76. CHAR\_INFO\* outputGrid;
77. SMALL\_RECT srctWriteRect;
78. void initDrawer();
79. void \_setFontSize(const WORD);
80. void \_destroy();
81. void \_swapBuf();
82. void \_draw();
83. void \_drawC();
84. };
85. FastPrinter::FastPrinter(DWORD x, DWORD y) :sizeX(x), sizeY(y) {
86. initDrawer();
87. }
88. FastPrinter::FastPrinter(DWORD x, DWORD y, WORD fontSize) : sizeX(x), sizeY(y) {
89. *// init with font size*
90. \_setFontSize(fontSize);
91. initDrawer();
92. }
93. FastPrinter::~FastPrinter() {
94. \_destroy();
95. }
96. void FastPrinter::setData(const char\* \_in\_data, const WORD\* \_in\_color) {
97. *// copy the data to inner buffer*
98. memcpy(dataGrid, \_in\_data, sizeX \* sizeY);
99. memcpy(colorGrid, \_in\_color, sizeX \* sizeY \* sizeof(WORD));
100. }
101. void FastPrinter::setData(const char\* \_in\_data, const WORD\* \_in\_color, SMALL\_RECT \_area) {
102. *// copy the data to the specified area*
103. SHORT row = (\_area.Right - \_area.Left);
104. for (WORD \_i = \_area.Top, i = 0; \_i < \_area.Bottom; \_i++, i++) {
105. memcpy(dataGrid + (\_i \* sizeX + \_area.Left), \_in\_data + (i \* row), row);
106. memcpy(colorGrid + (\_i \* sizeX + \_area.Left), \_in\_color + (i \* row), row \* sizeof(WORD));
107. }
108. }
109. void FastPrinter::setRect(SMALL\_RECT \_area, const char \_val, const WORD \_color) {
110. *// draw a hollow rectangle*
111. for (WORD i = \_area.Left; i < \_area.Right; i++) {
112. dataGrid[\_area.Top \* sizeX + i] = \_val;
113. dataGrid[(\_area.Bottom - 1) \* sizeX + i] = \_val;
114. colorGrid[\_area.Top \* sizeX + i] = \_color;
115. colorGrid[(\_area.Bottom - 1) \* sizeX + i] = \_color;
116. }
117. for (WORD i = \_area.Top; i < \_area.Bottom; i++) {
118. dataGrid[i \* sizeX + \_area.Left] = \_val;
119. dataGrid[i \* sizeX + \_area.Right - 1] = \_val;
120. colorGrid[i \* sizeX + \_area.Left] = \_color;
121. colorGrid[i \* sizeX + \_area.Right - 1] = \_color;
122. }
123. }
124. void FastPrinter::fillRect(SMALL\_RECT \_area, const char \_val, const WORD \_color) {
125. *// draw a solid rectangle*
126. SHORT row = (\_area.Right - \_area.Left);
127. for (WORD \_i = \_area.Top, i = 0; \_i < \_area.Bottom; \_i++, i++) {
128. memset(dataGrid + (\_i \* sizeX + \_area.Left), \_val, row);
129. for (WORD \_j = \_area.Left; \_j < \_area.Right; \_j++) {
130. colorGrid[\_i \* sizeX + \_j] = \_color;
131. }
132. }
133. }
134. void FastPrinter::setText(COORD \_pos, const char\* \_val, const WORD \_color, const WORD len) {
135. *// print text with position and color*
136. *// Note: try not to set text with '\n'*
137. memcpy(dataGrid + (\_pos.Y \* sizeX + \_pos.X), \_val, len);
138. for (WORD i = \_pos.X; i < \_pos.X + len; i++) {
139. colorGrid[\_pos.Y \* sizeX + i] = \_color;
140. }
141. }
142. void FastPrinter::setText(COORD \_pos, const char\* \_val, const WORD \_color) {
143. *// print text with position and color but no len*
144. WORD len = (WORD)strlen(\_val);
145. memcpy(dataGrid + (\_pos.Y \* sizeX + \_pos.X), \_val, len);
146. for (WORD i = \_pos.X; i < \_pos.X + len; i++) {
147. colorGrid[\_pos.Y \* sizeX + i] = \_color;
148. }
149. }
150. void FastPrinter::setText(COORD \_pos, const char\* \_val) {
151. *// print text with position but no len*
152. WORD len = (WORD)strlen(\_val);
153. memcpy(dataGrid + (\_pos.Y \* sizeX + \_pos.X), \_val, len);
154. for (WORD i = \_pos.X; i < \_pos.X + len; i++) {
155. colorGrid[\_pos.Y \* sizeX + i] = fp\_color::f\_l\_white;
156. }
157. }
158. void FastPrinter::\_setFontSize(const WORD x) {
159. CONSOLE\_FONT\_INFOEX cfi;
160. cfi.cbSize = sizeof(cfi);
161. GetCurrentConsoleFontEx(GetStdHandle(STD\_OUTPUT\_HANDLE), FALSE, &cfi);
162. cfi.dwFontSize.X = 0;
163. cfi.dwFontSize.Y = x;
164. SetCurrentConsoleFontEx(GetStdHandle(STD\_OUTPUT\_HANDLE), FALSE, &cfi);
165. }
166. void FastPrinter::cleanSrceen() {
167. memset(dataGrid, 0, sizeX \* sizeY);
168. memset(colorGrid, 0, sizeX \* sizeY \* sizeof(WORD));
169. memset(outputGrid, 0, sizeX \* sizeY \* sizeof(CHAR\_INFO));
170. }
171. void FastPrinter::draw(bool withColor) {
172. *// flush the whole screen*
173. *// here is a higher performance on drawc without using double buffer*
174. if (withColor)\_drawC();
175. else {
176. \_draw();
177. \_swapBuf();
178. }
179. }
180. void FastPrinter::initDrawer() {
181. *// init the data buffer*
182. dataGrid = new char[sizeX \* sizeY];
183. memset(dataGrid, 0, sizeX \* sizeY);
184. colorGrid = new WORD[sizeX \* sizeY];
185. memset(colorGrid, 0, sizeX \* sizeY \* sizeof(WORD));
186. outputGrid = new CHAR\_INFO[sizeX \* sizeY];
187. memset(outputGrid, 0, sizeX \* sizeY \* sizeof(CHAR\_INFO));
188. *// set the draw area*
189. srctWriteRect.Top = 0;
190. srctWriteRect.Left = 0;
191. srctWriteRect.Bottom = (SHORT)(sizeY - 1);
192. srctWriteRect.Right = (SHORT)(sizeX - 1);
193. *// get font size*
194. CONSOLE\_FONT\_INFOEX cfi;
195. cfi.cbSize = sizeof(cfi);
196. GetCurrentConsoleFontEx(GetStdHandle(STD\_OUTPUT\_HANDLE), FALSE, &cfi);
197. *// load the external WinAPI Module*
198. typedef HWND(WINAPI \*PROCGETCONSOLEWINDOW)();
199. PROCGETCONSOLEWINDOW GetConsoleWindow;
200. HMODULE hKernel32 = GetModuleHandleA("kernel32");
201. GetConsoleWindow = (PROCGETCONSOLEWINDOW)GetProcAddress(hKernel32, "GetConsoleWindow");
202. *// get console window handle and move the window to the upper left*
203. HWND hwnd = GetConsoleWindow();
204. SetWindowPos(hwnd, HWND\_TOP, 0, 0, cfi.dwFontSize.X \* sizeX, cfi.dwFontSize.Y \* sizeY, 0);
205. *// resize the window*
206. char cmd\_buffer[32] = "mode con: cols=0000 lines=0000";
207. cmd\_buffer[15] = '0' + (sizeX / 1000 % 10);
208. cmd\_buffer[16] = '0' + (sizeX / 100 % 10);
209. cmd\_buffer[17] = '0' + (sizeX / 10 % 10);
210. cmd\_buffer[18] = '0' + sizeX % 10;
211. cmd\_buffer[26] = '0' + (sizeY / 1000 % 10);
212. cmd\_buffer[27] = '0' + (sizeY / 100 % 10);
213. cmd\_buffer[28] = '0' + (sizeY / 10 % 10);
214. cmd\_buffer[29] = '0' + sizeY % 10;
215. system(cmd\_buffer);
216. *// create output buffer*
217. hOutBuf = CreateConsoleScreenBuffer(
218. GENERIC\_WRITE | GENERIC\_READ,
219. FILE\_SHARE\_WRITE | FILE\_SHARE\_READ,
220. NULL,
221. CONSOLE\_TEXTMODE\_BUFFER,
222. NULL
223. );
224. hOutput = CreateConsoleScreenBuffer(
225. GENERIC\_WRITE | GENERIC\_READ,
226. FILE\_SHARE\_WRITE | FILE\_SHARE\_READ,
227. NULL,
228. CONSOLE\_TEXTMODE\_BUFFER,
229. NULL
230. );
231. *// invisible the cursor*
232. CONSOLE\_CURSOR\_INFO cci;
233. cci.bVisible = 0;
234. cci.dwSize = 1;
235. SetConsoleCursorInfo(hOutput, &cci);
236. SetConsoleCursorInfo(hOutBuf, &cci);
237. SetConsoleActiveScreenBuffer(hOutput);
238. }
239. void FastPrinter::\_destroy() {
240. *// clean up memory*
241. delete[] dataGrid;
242. delete[] colorGrid;
243. delete[] outputGrid;
244. CloseHandle(hOutBuf);
245. CloseHandle(hOutput);
246. }
247. void FastPrinter::\_swapBuf() {
248. *// core function: display after the data has been set*
249. hTmpBuf = hOutBuf;
250. hOutBuf = hOutput;
251. hOutput = hTmpBuf;
252. }
253. void FastPrinter::\_draw() {
254. for (DWORD i = 0; i < sizeY; i++) {
255. *// draw every line*
256. coordBufCoord.Y = (SHORT)i;
257. WriteConsoleOutputCharacterA(hOutput, dataGrid + (i \* sizeX), sizeX, coordBufCoord, &bytes);
258. }
259. SetConsoleActiveScreenBuffer(hOutput);
260. }
261. void FastPrinter::\_drawC() {
262. for (DWORD i = 0; i < sizeY; i++) {
263. for (DWORD j = 0; j < sizeX; j++) {
264. *// copy info to CHAR\_INFO struct*
265. *// this will draw with color*
266. outputGrid[i \* sizeX + j].Attributes = colorGrid[i \* sizeX + j];
267. outputGrid[i \* sizeX + j].Char.AsciiChar = dataGrid[i \* sizeX + j];
268. }
269. }
270. coordBufCoord.X = 0;
271. coordBufCoord.Y = 0;
272. coordBufSize.X = (SHORT)(sizeX);
273. coordBufSize.Y = (SHORT)(sizeY);
274. WriteConsoleOutputA(
275. hOutput,          *// screen buffer to write to*
276. outputGrid,       *// buffer to copy from*
277. coordBufSize,     *// col-row size of chiBuffer*
278. coordBufCoord,    *// top left src cell in chiBuffer*
279. &srctWriteRect);  *// dest. screen buffer rectangle*
280. *//SetConsoleActiveScreenBuffer(hOutput);*
281. }
282. */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
283. \*  TO-DO END                                                      \*
284. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
285. #endif

3.demo.cpp

1. #define  \_CRT\_SECURE\_NO\_WARNINGS
2. #pragma comment(lib, "winmm.lib")
3. #include "PicReader.h"
4. #include "FastPrinter.h"
5. #include "mmsystem.h"
6. #include <stdio.h>
7. const static char asciiStrength[15] = {
8. 'M','N','H','Q','$','O','C','?','7','>','!',':','-',';','.'
9. };
10. char get\_ascii(int sp) {
11. return asciiStrength[sp / 18];
12. }
13. *//这里定义高度*
14. int HEI = 60;
15. void draw\_(const char addr[]) {
16. PicReader imread;
17. BYTE \*data = nullptr;
18. UINT x, y;
19. imread.readPic(addr);
20. Array<int> test;
21. UINT xx = 0, yy = 0;
22. imread.getData(test, xx, yy);
23. int times = yy / HEI;
24. if (times == 0) {
25. HEI = yy;
26. times = 1;
27. }
28. int LEN = xx / times;
29. if (LEN > 190) {
30. times = xx / 190;
31. HEI = yy / times;
32. LEN = 190;
33. }
34. int temp;
35. FastPrinter printer(LEN, HEI);
36. char \*databuffer = new char[LEN\*HEI];
37. WORD \*colorbuffer = new WORD[LEN\*HEI];
38. printer.cleanSrceen();
39. for (int j = 0; j < LEN; j++) {
40. for (int i = 0; i < HEI; i++) {
41. temp = 0;
42. for (int ii = i \* times; ii < (i + 1)\*times; ii++) {
43. for (int jj = j \* times; jj < (j + 1)\*times; jj++) {
44. temp += test[jj][ii];
45. }
46. }
47. temp = (temp + times \* times / 2) / (times\*times);
48. databuffer[i\*LEN + j] = get\_ascii(temp);
49. colorbuffer[i\*LEN + j] = fp\_color::b\_white | fp\_color::f\_black;
50. }
51. }
52. printer.setData(databuffer, colorbuffer);
53. printer.draw(true);
54. getchar();
55. HEI = 60;
56. }
57. const int NUM = 1011;
58. struct the\_id {
59. char \*databuffer;
60. }ids[NUM];
61. void video() {
62. PicReader imread;
63. BYTE \*data = nullptr;
64. UINT x, y;
65. Array<int> test;
66. UINT xx = 0, yy = 0;
67. char addr[20] = "video\\sea 0001.jpg";
68. int times, HEI = 60, LEN, temp;
69. WORD \*colorbuffer;
70. imread.readPic((const char\*)addr);
71. imread.getData(test, xx, yy);
72. times = yy / HEI;
73. if (times == 0) {
74. HEI = yy;
75. times = 1;
76. }
77. LEN = xx / times;
78. if (LEN > 190) {
79. times = xx / 190;
80. HEI = yy / times;
81. LEN = 190;
82. }
83. colorbuffer = new WORD[LEN\*HEI];
84. for (int j = 0; j < LEN; j++) {
85. for (int i = 0; i < HEI; i++) {
86. colorbuffer[i\*LEN + j] = fp\_color::b\_white | fp\_color::f\_black;
87. }
88. }
89. FastPrinter printer(LEN, HEI);
90. for (int uhi = 1; uhi <= NUM; uhi++) {
91. addr[10] = '0' + uhi / 1000;
92. addr[11] = '0' + (uhi % 1000) / 100;
93. addr[12] = '0' + (uhi % 100) / 10;
94. addr[13] = '0' + uhi % 10;
95. imread.readPic((const char\*)addr);
96. imread.getData(test, xx, yy);
97. ids[uhi].databuffer = new char[LEN\*HEI];
98. for (int j = 0; j < LEN; j++) {
99. for (int i = 0; i < HEI; i++) {
100. temp = 0;
101. for (int ii = i \* times; ii < (i + 1)\*times; ii++) {
102. for (int jj = j \* times; jj < (j + 1)\*times; jj++) {
103. temp += test[jj][ii];
104. }
105. }
106. temp = (temp + times \* times / 2) / (times\*times);
107. ids[uhi].databuffer[i\*LEN + j] = get\_ascii(temp);
108. colorbuffer[i\*LEN + j] = fp\_color::b\_white | fp\_color::f\_black;
109. }
110. }
111. HEI = 60;
112. }
113. printer.cleanSrceen();
114. COORD textXY;
115. textXY.X = 10;
116. textXY.Y = 10;
117. printer.setText(textXY, "按下回车开始播放");
118. printer.draw(true);
119. getchar();
120. DWORD tic, tic2;
121. tic = GetTickCount();
122. PlaySound("sound\\sea.wav", NULL, SND\_FILENAME | SND\_ASYNC);
123. for (int uhi = 1; uhi <= NUM; uhi++) {
124. for (;;) {
125. tic2 = GetTickCount();
126. if (tic2 - tic >= 40) {
127. tic += 40;
128. printer.cleanSrceen();
129. printer.setData(ids[uhi].databuffer, colorbuffer);
130. printer.draw(true);
131. break;
132. }
133. }
134. }
135. }
136. int main() {
137. char ccc;
139. while (true) {
140. printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 高 程 大 作 业 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
141. printf(" \* 请输入您想要查看的内容编号：                                   \* \n");
142. printf(" \* a) airplane.jpg       b) baboon.jpg         c) barbara.jpg     \* \n");
143. printf(" \* d) cameraman.jpg      e) compa.png          f) goldhill.jpg    \* \n");
144. printf(" \* g) lena.jpg           h) lena1.jpg          i) milkdrop.jpg    \* \n");
145. printf(" \* j) peppers.jpg        k) woman.jpg          l) sea.mp4         \* \n");
146. printf(" \*                       q)   退出                                \* \n");
147. printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1950000 一二三 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");
148. scanf("%c", &ccc);
149. getchar();
150. switch (ccc) {
151. case 'a':
152. draw\_("classic\_picture\\airplane.jpg");
153. break;
154. case 'b':
155. draw\_("classic\_picture\\baboon.jpg");
156. break;
157. case 'c':
158. draw\_("classic\_picture\\barbara.jpg");
159. break;
160. case 'd':
161. draw\_("classic\_picture\\cameraman.jpg");
162. break;
163. case 'e':
164. draw\_("classic\_picture\\compa.png");
165. break;
166. case 'f':
167. draw\_("classic\_picture\\goldhill.jpg");
168. break;
169. case 'g':
170. draw\_("classic\_picture\\lena.jpg");
171. break;
172. case 'h':
173. draw\_("classic\_picture\\lena1.jpg");
174. break;
175. case 'i':
176. draw\_("classic\_picture\\milkdrop.jpg");
177. break;
178. case 'j':
179. draw\_("classic\_picture\\peppers.jpg");
180. break;
181. case 'k':
182. draw\_("classic\_picture\\woman.jpg");
183. break;
184. case 'l':
185. video();
186. break;
187. }
188. if (ccc == 'q') {
189. break;
190. }
191. }
192. return 0;
193. }