同济大学电子信息学院
高级程序语言设计——字符画实验报告

学　　号	1750562
姓　　名	张博
专　　业	信息安全一班
授课老师	陈宇飞
日　　期	2019 年 6 月 5 日

一、设计思路与功能描述

总体和各部分的设计思路：

由于主要操作都依赖于 Array 数组进行，故创建了 Array 类，但 Array 类并不仅仅用作数组，图片存储的本质也是像素值数组，Array 类中封装多个函数用于图像像素读取、存储、矩阵运算、灰度值转 ASCII 码等多项操作。对于图片转字符画的设计思路为：先读取图像并转存每个像素的灰度值至 Array 类对象的数据成员 pic，随后根据图像大小决定图像是否压缩及压缩的程度，再根据灰度值映射到对应的 ASCII 码后输出。对于字符画视频的设计思路是：先将视频转为多个 JPG 图片，随后为每张图片计算灰度值和对应字符的映射，最后调节绘图的间歇时间达到绘制与原视频同步，并通过系统 API 播放音频。

Array 类数据成员和成员函数的功能描述：

```
    BYTE *pic;              //存储像素值、灰度值、映射后的 ASCII 码
    int size_x;             //矩阵规模
    int size_y;
    int size_z;
    Array(int a)        //构造函数：一维
    Array(int a, int b)         //构造函数：二维
    Array(int x, int y, int z) //构造函数：三维
    Array(BYTE *, unsigned int x, unsigned int y);        //构造函数：传入图像像素值
和图像长宽并直接计算各像素灰度值
    int at(int x, int y)                              //二维访问
    int at(int x, int y, int z)                           //三维访问
    void print();                                     //打印图片
    void print_player(FastPrinter &);                 //针对字符画视频的打印图片
    void to_ascii();                        //灰度值转 ASCII 码
    void to_ascii_player();                     //针对字符画视频的灰度值转 ASCII 码
    void add(Array &b);                         //矩阵相加
    void sub(Array &b);                         //矩阵相减
    void mul(Array &b);                         //矩阵点乘
    void div(Array &b);                         //矩阵点除
    void add(int a);
    void sub(int b);
    void mul(int a);
    void div(int b);
    void reshape(int a, int b, int c);          //reshape 操作
    BYTE* get_content();                    //返回指针并指向矩阵元数据
```

二、遇到的问题及对策

①：打印正方形图片时显示正常，长方形图片时出现乱码

解决：传参时，x 代表的是图片的长，y 代表的是图片的宽，故当需访问数组内的 (i,j)位置时，对于一维数组 pic[],中应该填入的值为：i*size_y+j

②：压缩图片导致图片显示异常

解决：压缩时应选择将该像素的像素平均值修改为附近 4×4 或者 2×2 的正方形中包含像素点的灰度值的平均值。

③：在展示字符画视频中反复闪烁控制台，且图片展示闪过

解决：不能像打印单个图片一样使用直接调用 print()，因为这样展示一个图片之后 FastPrinter 反复调用析构函数，导致控制台闪烁。解决方法是创建了一个新的函数成员，FastPrinter 对象在 print_player 函数外实现只调用一次析构函数。

三、心得体会及建议

通过这次大作业，更加了解了面向对象设计的优点。可以通过多个类的封装，将大工程进行子模块的划分，降低难度实现效果。加深了对字符画的理解，同时学会了使用视频剪辑软件对文件格式进行处理。重要的是通过对 FastPrinter 和 PicReader 的研究，让我觉得写出来这两个类的助教实在是强我太多！！！根本不是一个 level! FastPrinter 和 PicReader 中调用 WinAPI 有关 Console 绘制的底层函数开阔了我的视野，小菜鸡我都没有见过这些个 API！我觉得这次大作业如果让自己去实现的话就.....，能够实现出来还是站在了”巨人的肩膀上”。读取数据和打印数据都是基于两个文件提供的类实现。

没有什么好的建议，就希望以后我也能写出 FastPrinter 和 PicReader 这样的类吧！

四、源代码

my.cpp：

```
#pragma comment(lib,"winmm.lib")
#include "PicReader.h"
#include "FastPrinter.h"
#include <stdio.h>
#include<iostream>

using namespace std;


class Array
{
private:
    BYTE *pic;
public:
    int size_x;
    int size_y;
    int size_z;
    Array(int a)
    {
```

```
    pic = new BYTE[a];

    size_x = a;
    size_y = 0;
    size_z = 0;
}
Array(int a, int b) {
    pic = new BYTE[a*b];

    size_x = a;
    size_y = b;
    size_z = 0;
}
Array(int x, int y, int z)
{
    pic = new BYTE[x*y*z];

    size_x = x;
    size_y = y;
    size_z = z;
}
Array(BYTE *, unsigned int x, unsigned int y);
int at(int x, int y)
{
    int temp = x * size_x + y;
    return temp;
}
int at(int x, int y, int z)
{
    int temp = x * size_x + y * size_y + z;
    return temp;
}

void print();
void print_player(FastPrinter &);

void to_ascii();
void to_ascii_player();
void add(Array &b);                     //矩阵相加
void sub(Array &b);                     //矩阵相减
void mul(Array &b);                     //矩阵点乘
void div(Array &b);                     //矩阵点除
void add(int a);
void sub(int b);
```

```
        void mul(int a);
        void div(int b);
        void reshape(int a, int b, int c);              //reshape 操作
        BYTE* get_content();
};
void Array::to_ascii()
{
        char asciiStrength[] = { 'M','N','H','Q','$','O','C','?','7','>','!',':','-',';','.' };
        if (size_x <= 100)
        {
                BYTE    *p = pic;
                //BYTE temp;
                for (int i = 0; i < size_x * size_y; i++)
                {
                        *(p) = asciiStrength[(*p) / 18];
                        p++;
                }
        }
        else    if (size_x < 400)
        {
                /*
                if (x >= 400)
                        size_x = x / 2;
                if (y > 400)
                        size_y = y / 2;
                        */

                BYTE gray[4];
                BYTE *p2, *head;
                BYTE temp;
                head = new BYTE[size_x*size_y];
                p2 = head;
                for (int i = 0; i < size_y - 1; i += 2)
                {
                        for (int j = 0; j < size_x - 1; j += 2)
                        {
                                gray[0] = pic[this->at(i, j)];
                                gray[1] = pic[this->at(i, j + 1)];
                                gray[2] = pic[this->at(i + 1, j)];
                                gray[3] = pic[this->at(i + 1, j + 1)];
                                temp = (gray[0] + gray[1] + gray[2] + gray[3]) / (4);
                                *(p2++) = asciiStrength[temp / 18];
                        }
                }
```

```
		size_x /= 2;
		size_y /= 2;
		delete[]pic;
		pic = head;
	}
	else
	{
		BYTE gray[16];
		BYTE *p2, *head;
		BYTE temp;
		int m = 0;
		head = new BYTE[size_x*size_y];
		p2 = head;
		for (int i = 0; i < size_y - 3; i += 4)
		{
			for (int j = 0; j < size_x - 3; j += 4)
			{
				m = 0;
				gray[0] = pic[this->at(i, j)];
				gray[1] = pic[this->at(i, j + 1)];
				gray[2] = pic[this->at(i, j + 2)];
				gray[3] = pic[this->at(i + 1, j)];
				gray[4] = pic[this->at(i + 1, j + 1)];
				gray[5] = pic[this->at(i + 1, j + 2)];
				gray[6] = pic[this->at(i + 2, j)];
				gray[7] = pic[this->at(i + 2, j + 1)];
				gray[8] = pic[this->at(i + 2, j + 2)];
				gray[9] = pic[this->at(i, j + 3)];
				gray[10] = pic[this->at(i + 1, j + 3)];
				gray[11] = pic[this->at(i + 2, j + 3)];
				gray[12] = pic[this->at(i + 3, j + 3)];
				gray[13] = pic[this->at(i + 3, j)];
				gray[14] = pic[this->at(i + 3, j + 1)];
				gray[15] = pic[this->at(i + 3, j + 2)];
				for (int k = 0; k < 16; k++)
					m += gray[k];
				temp = m / 16;
				*(p2++) = asciiStrength[temp / 18];
			}
		}
		size_x /= 4;
		size_y /= 4;
		delete[]pic;
		pic = head;
```

```
        }
    }
    void Array::to_ascii_player()
    {
        char asciiStrength[] = { 'M','N','H','Q','$','O','C','?','7','>','!',':','-',';','.' };
        BYTE gray[4];
        BYTE *p2, *head;
        BYTE temp;
        head = new BYTE[size_x*size_y];
        p2 = head;
        for (int i = 0; i < size_y - 1; i += 2)
        {
            for (int j = 0; j < size_x - 1; j += 2)
            {
                gray[0] = pic[this->at(i, j)];
                gray[1] = pic[this->at(i, j + 1)];
                gray[2] = pic[this->at(i + 1, j)];
                gray[3] = pic[this->at(i + 1, j + 1)];
                temp = (gray[0] + gray[1] + gray[2] + gray[3]) / (4);
                *(p2++) = asciiStrength[temp / 18];
            }
        }
        size_x /= 2;
        size_y /= 2;
        delete[]pic;
        pic = head;
    }
    Array::Array(BYTE *data, unsigned int x, unsigned int y)                //存储灰度
值
    {
        size_x = x;
        size_y = y;
        pic = new BYTE[x*y];

        BYTE gray, *p = pic;
        BYTE temp;
        for (DWORD i = 0; i < x * y * 4; i += 4)
        {
            gray = (data[i] * 299 + data[i + 1] * 587 + data[i + 2] * 114 + 500) /
1000;
            temp = gray;
            *(p++) = temp;
        }
        delete[] data;
        data = nullptr;
    }
    void Array::add(Array &b)
    {

        if (size_x != b.size_x || size_y != b.size_y || size_z != b.size_z)
        {
            cout << "不可相加！ \n";
            return;
        }
        else
        {
            int dim = 0;
            dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
            switch (dim)
            {
            case 1:
                for (int i = 0; i < size_x; i++)
                {
                    pic[i] += b.pic[i];
                }
                break;
            case 2:
                for (int i = 0; i < size_x; i++)
                    for (int j = 0; j < size_y; j++)
                        pic[this->at(i, j)] += b.pic[b.at(i, j)];
                break;
            case 3:
                for (int i = 0; i < size_x; i++)
                    for (int j = 0; j < size_y; j++)
                        for (int k = 0; k < size_z; k++)
                            pic[this->at(i, j, k)] += b.pic[b.at(i, j, k)];
                break;
            default:break;
            }
        }

    }
    void Array::add(int n)
    {
        int dim = 0;
        dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
        switch (dim)
        {
        case 1:
            for (int i = 0; i < size_x; i++)
            {
                pic[i] += n;
            }
            break;
        case 2:
            for (int i = 0; i < size_x; i++)
                for (int j = 0; j < size_y; j++)
                    pic[this->at(i, j)] += n;
            break;
        case 3:
            for (int i = 0; i < size_x; i++)
                for (int j = 0; j < size_y; j++)
                    for (int k = 0; k < size_z; k++)
                        pic[this->at(i, j, k)] += n;
            break;
        default:break;
        }
}

void Array::sub(Array &b)
{

        if (size_x != b.size_x || size_y != b.size_y || size_z != b.size_z)
        {
            cout << "不可相加！ \n";
            return;
        }
        else
        {
            int dim = 0;
            dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
            switch (dim)
            {
            case 1:
                for (int i = 0; i < size_x; i++)
                {
                    pic[i] -= b.pic[i];
                }
                break;
            case 2:
                for (int i = 0; i < size_x; i++)
                    for (int j = 0; j < size_y; j++)
                    pic[this->at(i, j)] -= b.pic[b.at(i, j)];
                break;
            case 3:
                for (int i = 0; i < size_x; i++)
                    for (int j = 0; j < size_y; j++)
                        for (int k = 0; k < size_z; k++)
                            pic[this->at(i, j, k)] -= b.pic[b.at(i, j, k)];
                break;
            default:break;
            }
        }

}
void Array::sub(int n)
{
    int dim = 0;
    dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
    switch (dim)
    {
    case 1:
        for (int i = 0; i < size_x; i++)
        {
            pic[i] -= n;
        }
        break;
    case 2:
        for (int i = 0; i < size_x; i++)
            for (int j = 0; j < size_y; j++)
                pic[this->at(i, j)] -= n;
        break;
    case 3:
        for (int i = 0; i < size_x; i++)
            for (int j = 0; j < size_y; j++)
                for (int k = 0; k < size_z; k++)
                    pic[this->at(i, j, k)] -= n;
        break;
    default:break;
    }
}

void Array::mul(Array &b)
{

    if (size_x != b.size_x || size_y != b.size_y || size_z != b.size_z)
```

```
        {
            cout << "不可相加！ \n";
            return;
        }
        else
        {
            int dim = 0;
            dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
            switch (dim)
            {
            case 1:
                for (int i = 0; i < size_x; i++)
                {
                    pic[i] *= b.pic[i];
                }
                break;
            case 2:
                for (int i = 0; i < size_x; i++)
                    for (int j = 0; j < size_y; j++)
                        pic[this->at(i, j)] *= b.pic[b.at(i, j)];
                break;
            case 3:
                for (int i = 0; i < size_x; i++)
                    for (int j = 0; j < size_y; j++)
                        for (int k = 0; k < size_z; k++)
                            pic[this->at(i, j, k)] *= b.pic[b.at(i, j, k)];
                break;
            default:break;
            }
        }

    }
    void Array::mul(int n)
    {
        int dim = 0;
        dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
        switch (dim)
        {
        case 1:
            for (int i = 0; i < size_x; i++)
            {
                pic[i] *= n;
            }
            break;
```

```
	case 2:
		for (int i = 0; i < size_x; i++)
			for (int j = 0; j < size_y; j++)
				pic[this->at(i, j)] *= n;
		break;
	case 3:
		for (int i = 0; i < size_x; i++)
			for (int j = 0; j < size_y; j++)
				for (int k = 0; k < size_z; k++)
					pic[this->at(i, j, k)] *= n;
		break;
	default:break;
	}
}

void Array::div(Array &b)
{

	if (size_x != b.size_x || size_y != b.size_y || size_z != b.size_z)
	{
		cout << "不可相加！ \n";
		return;
	}
	else
	{
		int dim = 0;
		dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
		switch (dim)
		{
		case 1:
			for (int i = 0; i < size_x; i++)
			{
				pic[i] /= b.pic[i];
			}
			break;
		case 2:
			for (int i = 0; i < size_x; i++)
				for (int j = 0; j < size_y; j++)
					pic[this->at(i, j)] /= b.pic[b.at(i, j)];
			break;
		case 3:
			for (int i = 0; i < size_x; i++)
				for (int j = 0; j < size_y; j++)
					for (int k = 0; k < size_z; k++)
```

```
                        pic[this->at(i, j, k)] /= b.pic[b.at(i, j, k)];
                break;
            default:break;
            }
        }

}
void Array::div(int n)
{
    int dim = 0;
    dim = (size_x != 0) + (size_y != 0) + (size_z != 0);
    switch (dim)
    {
    case 1:
        for (int i = 0; i < size_x; i++)
        {
            pic[i] /= n;
        }
        break;
    case 2:
        for (int i = 0; i < size_x; i++)
            for (int j = 0; j < size_y; j++)
                pic[this->at(i, j)] /= n;
        break;
    case 3:
        for (int i = 0; i < size_x; i++)
            for (int j = 0; j < size_y; j++)
                for (int k = 0; k < size_z; k++)
                    pic[this->at(i, j, k)] /= n;
        break;
    default:break;
    }
}

void Array::reshape(int a = 0, int b = 0, int c = 0)
{
    size_x = a;
    size_y = b;
    size_z = c;
}

BYTE* Array::get_content()
{
    return pic;
```

```
	}
	void Array::print() {
		//system("color F0");
		WORD t = 1;
		FastPrinter printer(size_x * 2, size_y, t);
		COORD textXY;
		char ch[3];

		printer.cleanSrceen();
		for (int i = 0; i < size_y; i++)
		{
			for (int j = 0; j < size_x; j += 1)
			{
				ch[0] = pic[this->at(i, j)];
				ch[1] = ch[0];
				ch[2] = '\0';
				textXY.X = 2 * j;
				textXY.Y = i;
				printer.setText(textXY, ch, (fp_color::f_black | fp_color::b_l_white),
2);
				//cout << ch<<ch ;
			}
		}
		printer.draw(true);
		getchar();
	}

	void Array::print_player(FastPrinter &printer) {

		char ch[3];
		COORD textXY;
		printer.cleanSrceen();
		for (int i = 0; i < size_y; i++)
		{
			for (int j = 0; j < size_x; j += 1)
			{
				ch[0] = pic[this->at(i, j)];
				ch[1] = ch[0];
				ch[2] = '\0';
				textXY.X = 2 * j;
				textXY.Y = i;
				printer.setText(textXY, ch, (fp_color::f_black | fp_color::b_l_white),
2);
				//cout << ch<<ch ;
			}
		}
		printer.draw(true);
	}
	void getnum(int num, char temp[])
	{

		temp[2] = num % 10 + '0';
		num /= 10;
		temp[1] = num % 10 + '0';
		num /= 10;
		temp[0] = num % 10 + '0';

	}

	void changename(char str[])
	{
		char temp[3];
		int k = 0, num = 0, p = 0;
		while (str[k] != '\0')
		{
			if (str[k] <= '9'&&str[k] >= '0')
			{
				num *= 10;
				num += str[k] - '0';
			}
			k++;
		}					//已读取当前值
		num++;
		getnum(num, temp);
		k = 0;
		while (str[k] != '\0')
		{
			if (str[k] <= '9'&&str[k] >= '0')
			{
				str[k] = temp[p++];
			}
			k++;
		}

	}

	void player()
	{
    PicReader imread;
    char picname[30] = "kunkun\\kunkun 001.jpg";
    BYTE *data = nullptr;
    UINT x, y;


    imread.readPic(picname);
    imread.testReader(data, x, y);


    Array ima(data, x, y);

    ima.to_ascii_player();

    WORD t = 1;
    FastPrinter printer(ima.size_x * 2, ima.size_y, t);
    PlaySoundA("kunkun\\2.wav", NULL, SND_FILENAME | SND_ASYNC);
    ima.print_player(printer);
    changename(picname);

    //Sleep(3);
    for (int i = 1; i < 152; i++)
    {
        imread.readPic(picname);
        imread.testReader(data, x, y);


        Array ima(data, x, y);

        ima.to_ascii_player();

        ima.print_player(printer);
        changename(picname);

    }
}int main()
{

    PicReader imread;
    BYTE *data = nullptr;
    UINT x, y;

    imread.readPic("classic_picture\\lena.jpg");
```

```
        imread.testReader(data, x, y);

        Array ima(data, x, y);
        ima.to_ascii();
        ima.print();
        getchar();

        player();           //5 秒舞蹈欣赏
        return 0;
    }
```

FastPrinter.h:

```
/***************************************************************
* ！注意！                                                      *
* 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   *
* 速绘制你想要的输出，效率比 printf+cls 高出很多。                      *
* 函数使用详见 demo.cpp 中的几个示例。                                *
***************************************************************/
#ifndef FAST_PRINTER_H
#define FAST_PRINTER_H
#include <windows.h>

/***************************************************************
*  TO-DO:                                                     *
*                                                             *
*  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  *
*  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    *
*  一个更高效的。                                                *
*                                                             *
***************************************************************/

namespace fp_color {
    // f is the foreground color b is the background color
    // console color format: (f | b)
    const SHORT f_black = 0;
    const SHORT f_blue = 0x0001;
    const SHORT f_green = 0x0002;
    const SHORT f_aqua = 0x0003;
    const SHORT f_red = 0x0004;
    const SHORT f_purple = 0x0005;
    const SHORT f_yellow = 0x0006;
    const SHORT f_white = 0x0007;
    const SHORT f_gray = 0x0008;
    const SHORT f_l_blue = 0x0009;
    const SHORT f_l_green = 0x000A;
    const SHORT f_l_aqua = 0x000B;
    const SHORT f_l_red = 0x000C;
    const SHORT f_l_purple = 0x000D;
    const SHORT f_l_yellow = 0x000E;
    const SHORT f_l_white = 0x000F;

    const SHORT b_black = 0;
    const SHORT b_blue = 0x0010;
    const SHORT b_green = 0x0020;
    const SHORT b_aqua = 0x0030;
    const SHORT b_red = 0x0040;
    const SHORT b_purple = 0x0050;
    const SHORT b_yellow = 0x0060;
    const SHORT b_white = 0x0070;
    const SHORT b_gray = 0x0080;
    const SHORT b_l_blue = 0x0090;
    const SHORT b_l_green = 0x00A0;
    const SHORT b_l_aqua = 0x00B0;
    const SHORT b_l_red = 0x00C0;
    const SHORT b_l_purple = 0x00D0;
    const SHORT b_l_yellow = 0x00E0;
    const SHORT b_l_white = 0x00F0;
}

class FastPrinter {
public:
    FastPrinter(DWORD, DWORD);
    FastPrinter(DWORD, DWORD, WORD);
    ~FastPrinter();

    void setData(const char*, const WORD*);
    void setData(const char*, const WORD*, SMALL_RECT);
    void setRect(SMALL_RECT, const char, const WORD);
    void fillRect(SMALL_RECT, const char, const WORD);
    void setText(COORD, const char*, const WORD, const WORD);
    void setText(COORD, const char*, const WORD);
    void setText(COORD, const char*);

    void cleanSrceen();
    void draw(bool);
private:
    HANDLE hOutput, hOutBuf, hTmpBuf;
    COORD coordBufSize;
    COORD coordBufCoord;
    DWORD bytes = 0;
    DWORD sizeX, sizeY;

    char* dataGrid;
    WORD* colorGrid;
    CHAR_INFO* outputGrid;
    SMALL_RECT srctWriteRect;

    void initDrawer();
    void _setFontSize(const WORD);
    void _destroy();

    void _swapBuf();
    void _draw();
    void _drawC();
};

FastPrinter::FastPrinter(DWORD x, DWORD y) :sizeX(x), sizeY(y) {
    initDrawer();
}

FastPrinter::FastPrinter(DWORD x, DWORD y, WORD fontSize) : sizeX(x), sizeY(y) {
    // init with font size
    _setFontSize(fontSize);
    initDrawer();
}

FastPrinter::~FastPrinter() {
    _destroy();
}

void FastPrinter::setData(const char* _in_data, const WORD* _in_color) {
    // copy the data to inner buffer
    memcpy(dataGrid, _in_data, sizeX * sizeY);
    memcpy(colorGrid, _in_color, sizeX * sizeY * sizeof(WORD));
}

void FastPrinter::setData(const char* _in_data, const WORD* _in_color, SMALL_RECT _area)
{
    // copy the data to the specified area
    SHORT row = (_area.Right - _area.Left);
    for (WORD _i = _area.Top, i = 0; _i < _area.Bottom; _i++, i++) {
        memcpy(dataGrid + (_i * sizeX + _area.Left), _in_data + (i * row), row);
        memcpy(colorGrid + (_i * sizeX + _area.Left), _in_color + (i * row), row *
sizeof(WORD));
    }
}

void FastPrinter::setRect(SMALL_RECT _area, const char _val, const WORD _color) {
    // draw a hollow rectangle
    for (WORD i = _area.Left; i < _area.Right; i++) {
        dataGrid[_area.Top * sizeX + i] = _val;
        dataGrid[(_area.Bottom - 1) * sizeX + i] = _val;

        colorGrid[_area.Top * sizeX + i] = _color;
        colorGrid[(_area.Bottom - 1) * sizeX + i] = _color;
    }

    for (WORD i = _area.Top; i < _area.Bottom; i++) {
        dataGrid[i * sizeX + _area.Left] = _val;
        dataGrid[i * sizeX + _area.Right - 1] = _val;

        colorGrid[i * sizeX + _area.Left] = _color;
        colorGrid[i * sizeX + _area.Right - 1] = _color;
    }
}

void FastPrinter::fillRect(SMALL_RECT _area, const char _val, const WORD _color) {
    // draw a solid rectangle
    SHORT row = (_area.Right - _area.Left);
    for (WORD _i = _area.Top, i = 0; _i < _area.Bottom; _i++, i++) {
        memset(dataGrid + (_i * sizeX + _area.Left), _val, row);
        for (WORD _j = _area.Left; _j < _area.Right; _j++) {
            colorGrid[_i * sizeX + _j] = _color;
        }
    }
}

void FastPrinter::setText(COORD _pos, const char* _val, const WORD _color, const WORD len)
{
    // print text with position and color
    // Note: try not to set text with '\n'
    memcpy(dataGrid + (_pos.Y * sizeX + _pos.X), _val, len);
    for (WORD i = _pos.X; i < _pos.X + len; i++) {
        colorGrid[_pos.Y * sizeX + i] = _color;
    }
}
```

```
void FastPrinter::setText(COORD _pos, const char* _val, const WORD _color) {
    // print text with position and color but no len
    WORD len = (WORD)strlen(_val);
    memcpy(dataGrid + (_pos.Y * sizeX + _pos.X), _val, len);
    for (WORD i = _pos.X; i < _pos.X + len; i++) {
        colorGrid[_pos.Y * sizeX + i] = _color;
    }
}

void FastPrinter::setText(COORD _pos, const char* _val) {
    // print text with position but no len
    WORD len = (WORD)strlen(_val);
    memcpy(dataGrid + (_pos.Y * sizeX + _pos.X), _val, len);
    for (WORD i = _pos.X; i < _pos.X + len; i++) {
        colorGrid[_pos.Y * sizeX + i] = fp_color::f_l_white;
    }
}

void FastPrinter::_setFontSize(const WORD x) {
    CONSOLE_FONT_INFOEX cfi;
    cfi.cbSize = sizeof(cfi);
    GetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi);
    cfi.dwFontSize.X = 0;
    cfi.dwFontSize.Y = x;
    SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi);
}


void FastPrinter::cleanSrceen() {
    memset(dataGrid, 0, sizeX * sizeY);
    memset(colorGrid, 0, sizeX * sizeY * sizeof(WORD));
    memset(outputGrid, 0, sizeX * sizeY * sizeof(CHAR_INFO));
}

void FastPrinter::draw(bool withColor) {
    // flush the whole screen
    if (withColor)_drawC();
    else _draw();
    _swapBuf();
}

void FastPrinter::initDrawer() {
    // init the data buffer
    dataGrid = new char[sizeX * sizeY];
    memset(dataGrid, 0, sizeX * sizeY);

    colorGrid = new WORD[sizeX * sizeY];
    memset(colorGrid, 0, sizeX * sizeY * sizeof(WORD));

    outputGrid = new CHAR_INFO[sizeX * sizeY];
    memset(outputGrid, 0, sizeX * sizeY * sizeof(CHAR_INFO));

    // set the draw area
    srctWriteRect.Top = 0;
    srctWriteRect.Left = 0;
    srctWriteRect.Bottom = (SHORT)(sizeY - 1);
    srctWriteRect.Right = (SHORT)(sizeX - 1);

    // get font size
    CONSOLE_FONT_INFOEX cfi;
    cfi.cbSize = sizeof(cfi);
    GetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi);

    // load the external WinAPI Module
    typedef HWND(WINAPI *PROCGETCONSOLEWINDOW)();
    PROCGETCONSOLEWINDOW GetConsoleWindow;
    HMODULE hKernel32 = GetModuleHandleA("kernel32");
    GetConsoleWindow = (PROCGETCONSOLEWINDOW)GetProcAddress(hKernel32,
"GetConsoleWindow");

    // get console window handle and move the window to the upper left
    HWND hwnd = GetConsoleWindow();
    SetWindowPos(hwnd, HWND_TOP, 0, 0, cfi.dwFontSize.X * sizeX, cfi.dwFontSize.Y * sizeY,
0);

    // resize the window
    char cmd_buffer[32] = "mode con: cols=0000 lines=0000";
    cmd_buffer[15] = '0' + (sizeX / 1000 % 10);
    cmd_buffer[16] = '0' + (sizeX / 100 % 10);
    cmd_buffer[17] = '0' + (sizeX / 10 % 10);
    cmd_buffer[18] = '0' + sizeX % 10;

    cmd_buffer[26] = '0' + (sizeY / 1000 % 10);
    cmd_buffer[27] = '0' + (sizeY / 100 % 10);
    cmd_buffer[28] = '0' + (sizeY / 10 % 10);
    cmd_buffer[29] = '0' + sizeY % 10;
    system(cmd_buffer);

    // create output buffer
    hOutBuf = CreateConsoleScreenBuffer(
        GENERIC_WRITE | GENERIC_READ,
        FILE_SHARE_WRITE | FILE_SHARE_READ,
        NULL,
        CONSOLE_TEXTMODE_BUFFER,
        NULL
    );

    hOutput = CreateConsoleScreenBuffer(
        GENERIC_WRITE | GENERIC_READ,
        FILE_SHARE_WRITE | FILE_SHARE_READ,
        NULL,
        CONSOLE_TEXTMODE_BUFFER,
        NULL
    );

    // invisible the cursor
    CONSOLE_CURSOR_INFO cci;
    cci.bVisible = 0;
    cci.dwSize = 1;
    SetConsoleCursorInfo(hOutput, &cci);
    SetConsoleCursorInfo(hOutBuf, &cci);
}

void FastPrinter::_destroy() {
    // clean up memory
    delete[] dataGrid;
    delete[] colorGrid;
    delete[] outputGrid;

    CloseHandle(hOutBuf);
    CloseHandle(hOutput);
}

void FastPrinter::_swapBuf() {
    // core function: display after the data has been set
    hTmpBuf = hOutBuf;
    hOutBuf = hOutput;
    hOutput = hTmpBuf;
}
```

```
void FastPrinter::_draw() {
    for (DWORD i = 0; i < sizeY; i++) {
        // draw every line
        coordBufCoord.Y = (SHORT)i;
        WriteConsoleOutputCharacterA(hOutput, dataGrid + (i * sizeX), sizeX, coordBufCoord,
&bytes);
    }
    SetConsoleActiveScreenBuffer(hOutput);
}

void FastPrinter::_drawC() {
    for (DWORD i = 0; i < sizeY; i++) {
        for (DWORD j = 0; j < sizeX; j++) {
            // copy info to CHAR_INFO struct
            // this will draw with color
            outputGrid[i * sizeX + j].Attributes = colorGrid[i * sizeX + j];
            outputGrid[i * sizeX + j].Char.AsciiChar = dataGrid[i * sizeX + j];
        }
    }

    coordBufCoord.X = 0;
    coordBufCoord.Y = 0;
    coordBufSize.X = (SHORT)(sizeX);
    coordBufSize.Y = (SHORT)(sizeY);

    WriteConsoleOutputA(
        hOutput,          // screen buffer to write to
        outputGrid,       // buffer to copy from
        coordBufSize,     // col-row size of chiBuffer
        coordBufCoord,    // top left src cell in chiBuffer
        &srctWriteRect);  // dest. screen buffer rectangle
    SetConsoleActiveScreenBuffer(hOutput);
}
/******************************************************************
*  TO-DO END                                                     *
******************************************************************/
    #endif
```

PicReader.h:

```
/******************************************************************
* ! 注意!                                                          *
* 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  *
* 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  *
*     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，       *
* 请及时联系老师或助教！                                                    *
*     每一个 TO-DO 块以 TO-DO：说明 （TO-DO） END 结束，具体可看下方代码   *
*     readPic()函数为你封装了 WinAPI 中的方法，可以将图片读取为 RGBA 的    *
* 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    *
* 目的。                                                                   *
*     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    *
* 成你实现的 Array 类。                                                     *
*     testReader()是 demo 中提供读取数据的其中一个思路。                   *
******************************************************************/
#ifndef PIC_READER_H
#define PIC_READER_H

#include <windows.h>
#include <wincodec.h>
#include <commdlg.h>

template <typename T>
inline void SafeRelease(T *&p) {
    if (nullptr != p) {
        p->Release();
        p = nullptr;
    }
}

class PicReader {
public:
    PicReader();
    ~PicReader();
    void readPic(LPCSTR);
    void getData();
    void testReader(BYTE *&,UINT &, UINT &);
private:
    void init();
    bool checkHR(HRESULT);
    void quitWithError(LPCSTR);

    HWND                    hWnd;           //表示窗口
    HANDLE                  hFile;          //通用句柄，表示对象
    IWICImagingFactory     *m_pIWICFactory;
    IWICFormatConverter    *m_pConvertedSourceBitmap;

    /*TO-DO：这里可能会增加你需要的内部成员 END*/
};
```

```cpp
PicReader::PicReader() : m_pConvertedSourceBitmap(nullptr), m_pIWICFactory(nullptr) {
    init();
}

PicReader::~PicReader() {
    if (hFile != NULL) CloseHandle(hFile);
    SafeRelease(m_pConvertedSourceBitmap);
    SafeRelease(m_pIWICFactory);
    CoUninitialize();
}

bool PicReader::checkHR(HRESULT hr) {
    return (hr < 0);
}

void PicReader::quitWithError(LPCSTR message) {
    MessageBoxA(hWnd, message, "Application Error", MB_ICONEXCLAMATION | MB_OK);
    quick_exit(0xffffffff);
}

void PicReader::init() {
    hWnd = GetForegroundWindow();

    // Enables the terminate-on-corruption feature.
    HeapSetInformation(nullptr, HeapEnableTerminationOnCorruption, nullptr, 0);

    HRESULT hr = S_OK;

    //Init the WIC
    hr = CoInitializeEx(nullptr, COINIT_APARTMENTTHREADED | COINIT_DISABLE_OLE1DDE);

    // Create WIC factory
    hr = CoCreateInstance(
        CLSID_WICImagingFactory,
        nullptr,
        CLSCTX_INPROC_SERVER,
        IID_PPV_ARGS(&m_pIWICFactory)
    );

    // Throw error if create factor failed
    if (checkHR(hr)) { quitWithError("Init Reader Failed"); }
}

void PicReader::readPic(LPCSTR fileName) {
    HRESULT hr = S_OK;

    // Create a File Handle (WinAPI method not std c)
    if (hFile != NULL) CloseHandle(hFile);
    hFile = CreateFileA(fileName, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL, NULL);
    if (GetLastError() == ERROR_FILE_NOT_FOUND) {
        quitWithError("Cannot find such file, please retry or check the access");
    }

    // Create a decoder
    IWICBitmapDecoder *pDecoder = nullptr;
    hr = m_pIWICFactory->CreateDecoderFromFileHandle((ULONG_PTR)hFile, nullptr,
WICDecodeMetadataCacheOnDemand, &pDecoder);
    if (checkHR(hr)) { quitWithError("Create Decoder Failed"); }

    // Retrieve the first frame of the image from the decoder
    IWICBitmapFrameDecode *pFrame = nullptr;
    hr = pDecoder->GetFrame(0, &pFrame);
    if (checkHR(hr)) { quitWithError("Get Frame Failed"); }

    // Format convert the frame to 32bppRGBA
    SafeRelease(m_pConvertedSourceBitmap);
    hr = m_pIWICFactory->CreateFormatConverter(&m_pConvertedSourceBitmap);
    if (checkHR(hr)) { quitWithError("Get Format Converter Failed"); }

    hr = m_pConvertedSourceBitmap->Initialize(pFrame, GUID_WICPixelFormat32bppRGBA,
WICBitmapDitherTypeNone, nullptr, 0.f, WICBitmapPaletteTypeCustom);
    if (checkHR(hr)) { quitWithError("Init Bitmap Failed"); }

    // Clean memory
    SafeRelease(pDecoder);
    SafeRelease(pFrame);
}

void PicReader::getData() {
    HRESULT hr = S_OK;

    // Get the size of Image
    UINT x, y;
    hr = m_pConvertedSourceBitmap->GetSize(&x, &y);
    if (checkHR(hr)) { quitWithError("Check Bitmap Size Failed"); }

    // Create the buffer of pixels, the type of BYTE is unsigned char
    BYTE *data;
    data = new BYTE[x * y * 4];
    memset(data, 0, x * y * 4);             //经常用于对新申请的地址初始化

    // Copy the pixels to the buffer
    UINT stride = x * 4;
    hr = m_pConvertedSourceBitmap->CopyPixels(nullptr, stride, x * y * 4, data);
    if (checkHR(hr)) { quitWithError("Copy Pixels Failed"); }

    /******************************************************************
    *  TO-DO:                                                         *
    *                                                                 *
    *  实现一个 Array 类，并将上面的 data 转存至你的 Array 内                *
    *                                                                 *
    *  数据说明：从 Bitmap Copy 出来的数据，每 4 个为一组代表一个像素          *
    *           数据为一个长度为图像的(长*宽*4)的一维数组                   *
    *           即数据排布为 R G B A R G B A R G B A.....                *
    *                                                                 *
    *  ！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！        *
    ******************************************************************/

    //Array ima(data, x, y);            //转存至 Array 内
    //ima.reshape(x, y, 4);             //reshape 为 长*宽*4 矩阵
    delete[] data;

    /******************************************************************
    *  TO-DO END                                                      *
    ******************************************************************/

    // Close the file handle
    CloseHandle(hFile);
    hFile = NULL;
}

void PicReader::testReader(BYTE* &_out, UINT& _x, UINT& _y){
    HRESULT hr = S_OK;

    // Get the size of Image
    UINT x, y;
    hr = m_pConvertedSourceBitmap->GetSize(&x, &y);
    if (checkHR(hr)) { quitWithError("Check Bitmap Size Failed"); }

    // Create the buffer of pixels, the type of BYTE is unsigned char
    BYTE *data;
    data = new BYTE[x * y * 4];
    memset(data, 0, x * y * 4);

    // Copy the pixels to the buffer
    UINT stride = x * 4;
    hr = m_pConvertedSourceBitmap->CopyPixels(nullptr, stride, x * y * 4, data);
    if (checkHR(hr)) { quitWithError("Copy Pixels Failed"); }


    _out = data; _x = x; _y = y;

    // Close the file handle
    CloseHandle(hFile);
    hFile = NULL;
}

    #endif
```