《字符画》大作业报告

班级：智能交通与车辆 14 班

学号：2152402

姓名：段婷婷

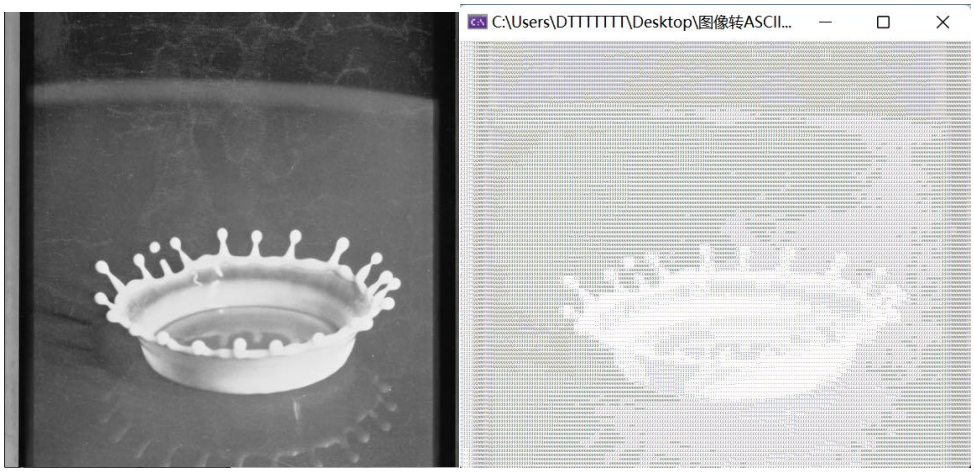
完成日期：2022.6.17

目录

一. 功能描述与设计思路

1. 功能描述:

将正方形图片转为 ASCII 的小程序，如下图所示:



2. 设计思路:

（1）实现 Array 类

1）包含的变量及其含义

int* data; //Array 中的数据

int index; //当前处于的下标

int shape[16];//每一维的大小

int totNum[16];//每一维包含的数据的数目

int axisNum; //总共有几维

int nowAxis; //现在在第几维

2）成员函数及其作用

Array(Args... args){}; //获取参数包，依照参数包构造对象。

Array(Array* arr) {}; //不完全的复制构造函数(说不完全,是因为不复制 data)

void reshape(Args... args){}; //即从一个一维矩阵 reshape 成二维或三维矩阵

void newData() {}; //给 data 动态申请 totNum[0]大小的空间

void copyData(Array& arr){};

//复制 arr 的 data 数据（与不复制 data 的构造函数相辅相成）

void releaseData() {}; //释放掉为 data 而动态申请的空间

Array operator[](int index){};

//此函数属于核心函数

是能够像使用多维数组那样方便地使用 Array 对象的关键

```
Array operator[](int index)
{
    // 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
    Array subArr(this);
    ++subArr.nowAxis;
    subArr.index = this->index + totNum[nowAxis] * index;
    subArr.data = data;
    return subArr;
}
```

Array& operator=(int data){}; //赋值，给 this->data[index]附 data 值

operator int() {}; //返回 data[index]

Array operator+(Array& arr){};
Array operator+(int Data){};
Array operator-(Array& arr){};
Array operator-(int Data){};
Array operator*(Array& arr){};
Array operator*(int Data){};
Array operator/(Array& arr){};
Array operator/(int Data){};

//以上六个函数实现矩阵的加减乘除的基本运算操作

(2) 图片转 ASCII

Step1: 获取图片的 RGB 信息(利用 PicReader 获取含此信息的 Array 对象)

Step2: 将图片的 RGB 信息转换成灰度值（公式如下）

$$Gray = (Red * 299 + Green * 587 + Blue * 114 + 500)/1000$$

Step3: 选择合适的小正方形大小, 每个小正方形压缩成一个像素点处理 (取灰度值的平均值)

Step4: 对于每个区间的灰度值选定字符来代替

Step5: 将字符连成字符串并输出至控制台（利用 FastPrinter）

二. 在实验过程中遇到的困难及解决方法

1. **困难**：如何实现 Array 中的重载[]?

解决方法：在重载[]的函数中，定义一个新的 Array 对象（复制构造），此对象与原对象**共享一段地址的 data**，修改新对象的 nowAxis 与 index，返回新对象即可。

2. **困难**：按照上一个困难的解决方法，新对象与原对象共享一段 data，如何释放动态申请的空间?

解决方法：析构函数设置为空，另外单独设置一个 void releaseData()函数来释放空间，在需要的时候显示调用即可。

3. **困难**：程序运行非常慢，发现是每次用[]时都会调用复制构造函数，动态申请空间。

解决方法：在复制构造函数中，不复制 data 值，也不给 data 值申请空间，这样调用[]时就不会在逐个值复制或动态申请空间上花费大量的不必要的时间了。另外单独写 void newData(); 和 void copyData(Array& arr); 函数，用来满足在复制构造时，需要申请空间或逐个复制 data 值的需求，必要时显示调用即可。

4. **困难**：字符画纵向被压扁，怎么解决？

原因：字符的高与宽之比为 2：1，一个像素点被一个矩形表示导致的。

解决方法：一个像素点用两个同样的字符表示。

三．心得体会

完成一个 Array 的类，可以像多维数组一样方便的使用，真的觉得好奇妙哇（emm 然后就词穷了，总之花心思从底层构造一些以前非常自然顺畅地用地东西真的是非常好的体验！）。

不过，最大的体会是，再也不敢赶大作业的 ddl 了！算是真正意义上的第一次赶大作业 ddl，ddl 前一天晚上才开始写的。总的过程非常非常非常痛苦，当你想要中途休整一下的时候会发现距离 ddl 还有 x 小时，距离完成还有 50%（哭）。总之是想写写不出，想停不敢停。~~（不过，也没有下个大作业了）~~ 所以，作业做得也是不能再粗糙了，还是很遗憾没有好好结尾的（都是自己的锅鸣）。鉴于答辩选择了这个大作业，所以考完考试肯定还是会好好完善拓展一下字符画的！

高程荣课程结束啦，回看起来还是感慨的。学会了很多新知识，完成了几个最初一头雾水的大作业，虽然不完美，但是总归都用心、花时间完成了，还是很有成就感的！在这门课上花了很多时间和精力，这样强度的训练也放大了我的很

多问题，逼我直面和改正（否则？否则就会写不完作业: (）。比如不愿意一步一步调试而是抱着侥幸心理，想到一个问题然后一通改最后发现还是不多*n 次，比如面对未知的东西总是想要逃避.....

在知识方面最大的收获，是对指针和类的掌握！在上高程荣之前，我完全不理解为什么会有指针的存在，我只觉得它是个除了把问题变复杂以外啥用没有的东西。在飞姐的反复强调下，我深刻地意识到了指针的关键性，并且在大量的练习中还算不错地掌握了指针。现在，我会在没有被要求的情况下自发地比较顺畅地使用指针，是我以前绝不会想到的。 最大最大的非知识类收获可能是，学习全新知识的勇气和能力！在没有老师/书籍/PPT 保姆式一条龙服务地教一个东西时，我需要自己查找资料、独立思考、提出问题并且尝试解决、与同学/老师沟通讨论问题，最后完成作业。这是学习其他课程时没有地体验，也是非常宝贵的经历！

（最后想说开学第一节课就想说的，飞姐真的好像我高中时候的教练，长相发型和声音都像，所以一直都觉得很亲切 hhh！ ）

总之，真的很感谢遇见高程荣、可爱的飞姐、助教以及一起努力的同学们！

四．源代码

1．源.cpp

```
#include"PicReader.h"
#include"FastPrinter.h"
using namespace std;

char asciiStrength[] = { 'M','N','H','Q','$','0','C','?','7','>','!',':','-',';','.' };
int tmp[200][200];

void func(const char* infile) {
    PicReader imread;
    imread.readPic(infile);
    UINT x, y;
    Array arr1 = imread.getData(y, x); //y 是宽度（横），x 是长度（竖）

    if (x != y) {
        if (x < y) y = x;
        else x = y;
    }
```

```
    Array arr2(x, y);
    for (int i = 0;i < x;i++) {
        for (int j = 0;j < y;j++) {
            arr2[i][j] = int((int)arr1[i][j][0] * 299 + (int)arr1[i][j][1] * 587 + (int)arr1[i][j][2] * 114
+ 500) / 1000; //计算灰度
        }
    }

    const int div = min(x,y) / 120 + 1;
    for (int i = 0;i < x / div;i++) {
        for (int j = 0;j < y / div;j++) {
            tmp[i][j] = 0;
            for (int ki = div * i;ki < div * i + div;ki++) {
                for (int kj = div * j;kj < div * j + div;kj++) {
                    tmp[i][j] += int(arr2[ki][kj]);
                }
            }
            tmp[i][j] = tmp[i][j] / (div * div);
        }
    }
    for (int i = 0;i < x / div;i++)
        for (int j = 0;j < y / div;j++) {
            arr2[i][j] = unsigned char(int(tmp[i][j]));
        }
    x /= div, y /= div;

    FastPrinter printer(y * 2, x, 3);
    printer.cleanSrceen();

    char* dataBuffer = new char[200 * 200];
    for (int i = 0;i < x;i++)
        for (int j = 0;j < y * 2;j += 2) {
            unsigned char asciiIndex = int(arr2[i][j / 2]) / 18;
            dataBuffer[i * y * 2 + j] = asciiStrength[asciiIndex];
            dataBuffer[i * y * 2 + j + 1] = asciiStrength[asciiIndex];
        }

    BYTE* frontColorBuffer = new BYTE[200 * 200 * 3];
    BYTE* backColorBuffer = new BYTE[200 * 200 * 3];

    for (int i = 0;i < x * y * 6;i++) {
        frontColorBuffer[i] = 0;
        backColorBuffer[i] = 255;
```

```
        }


        /*SMALL_RECT drawArea;
        drawArea.Left = 0;
        drawArea.Right = y*2-1;
        drawArea.Top = 0;
        drawArea.Bottom = x-1;*/

        printer.setData(dataBuffer, frontColorBuffer, backColorBuffer);
        //printer.setData(dataBuffer, frontColorBuffer, backColorBuffer, drawArea);
        printer.draw(true);

        getchar();

}

int main() {
        cout << "每输出一张字符画，回车跳转至下一张" << endl;
        cout << "按回车键开始" << endl;
        getchar();


        //func("picture\\compa.png");
        //func("picture\\goldhill.jpg");
        //func("picture\\barbara.jpg");

        func("picture\\airplane.jpg");
        func("picture\\baboon.jpg");
        func("picture\\cameraman.jpg");
        func("picture\\lena.jpg");
        func("picture\\milkdrop.jpg");
        func("picture\\peppers.jpg");
        func("picture\\woman.jpg");
        return 0;
}
```

2. Array.cpp

```
#pragma once
#include <iostream>
using namespace std;

class Array
{
```

```
public:
    template <typename... Args>
    Array(Args... args)
    {
        // 获取参数包大小并转换为 size_t 数组
        auto num = sizeof...(args);
        size_t list[] = { args... };

        index = 0;
        axisNum = num;
        nowAxis = 1;
        for (int i = 1;i <= num;i++)
            shape[i] = list[i - 1];
        totNum[num] = 1;
        for (int i = num - 1;i >= 0;i--)
            totNum[i] = shape[i + 1] * totNum[i + 1];
        data = new int[totNum[0]];
    }
    Array(Array* arr) {
        axisNum = arr->axisNum;
        nowAxis = arr->nowAxis;
        index = arr->index;
        for (int i = 0;i <= axisNum;i++) {
            totNum[i] = arr->totNum[i];
            shape[i] = arr->shape[i];
        }
    }
    ~Array() {
    }

    template <typename... Args>
    Array at(Args... args)
    {
        // 获取参数包大小并转换为 size_t 数组
        auto num = sizeof...(args);
        size_t list[] = { args... };
    }

    template <typename... Args>
    void reshape(Args... args)
    {
        // 获取参数包大小并转换为 size_t 数组
        auto num = sizeof...(args);
        size_t list[] = { args... };
```

```
        index = 0;
        axisNum = num;
        nowAxis = 1;
        for (int i = 1;i <= num;i++)
            shape[i] = list[i - 1];
        totNum[num] = 1;
        for (int i = num - 1;i >= 0;i--)
            totNum[i] = shape[i + 1] * totNum[i + 1];

}

int* get_content() { return data; }

void set(int value) {}

void copyData(Array& arr) {
        for (int i = 0;i < totNum[0];i++)
            data[i] = arr.data[i];
}

void newData() {
        data = new int[totNum[0]];
}

void releaseData() {
        delete[] data;
}

Array operator[](int index)
{
        // 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
        Array subArr(this);
        ++subArr.nowAxis;
        subArr.index = this->index + totNum[nowAxis] * index;
        subArr.data = data;
        return subArr;
}

Array& operator=(int data)
{
        //cout << "index=" << index << endl;
        this->data[index] = data;
        return *this;
```

```
}

operator int() {
	//cout << "index=" << index << endl;
	return data[index];
}

Array operator+(Array& arr) {
	Array ret(this);
	ret.newData();
	for (int i = 0;i < totNum[0];i++)
		ret.data[i] = data[i] + arr.data[i];
	return ret;
}

Array operator+(int Data) {
	Array ret(this);
	ret.newData();
	for (int i = 0;i < totNum[0];i++)
		ret.data[i] = data[i] + Data;
	return ret;
}

Array operator-(Array& arr) {
	Array ret(this);
	ret.newData();
	for (int i = 0;i < totNum[0];i++)
		ret.data[i] = data[i] - arr.data[i];
	return ret;
}

Array operator-(int Data) {
	Array ret(this);
	ret.newData();
	for (int i = 0;i < totNum[0];i++)
		ret.data[i] = data[i] - Data;
	return ret;
}

Array operator*(Array& arr) {
	Array ret(this);
	ret.newData();
	for (int i = 0;i < totNum[0];i++)
		ret.data[i] = data[i] * arr.data[i];
```

```
        return ret;
    }

    Array operator*(int Data) {
        Array ret(this);
        ret.newData();
        for (int i = 0;i < totNum[0];i++)
            ret.data[i] = data[i] * Data;
        return ret;
    }

    Array operator/(Array& arr) {
        Array ret(this);
        ret.newData();
        for (int i = 0;i < totNum[0];i++)
            if (arr.data[i] != 0) ret.data[i] = data[i] / arr.data[i];
        return ret;
    }

    Array operator/(int Data) {
        Array ret(this);
        ret.newData();
        for (int i = 0;i < totNum[0];i++)
            ret.data[i] = data[i] / Data;
        return ret;
    }


    int* data;
    int index; //当前处于的下标
    int shape[16];//每一维的大小
    int totNum[16];
    int axisNum; //总共有几维
    int nowAxis; //现在在第几维
};
```

3. PicReader.h

```
#pragma once
/*****************************************************************
* ！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>
#include"Array.cpp"

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

class PicReader {
public:
    PicReader();
    ~PicReader();

    void readPic(LPCSTR);
    Array getData(UINT&, UINT&);
    void testReader(BYTE*&, UINT&, UINT&);
private:
    void init();
    bool checkHR(HRESULT);
    void quitWithError(LPCSTR);

    HWND                    hWnd;
    HANDLE                  hFile;
    IWICImagingFactory* m_pIWICFactory;
    IWICFormatConverter* m_pConvertedSourceBitmap;

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

```
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);
}

Array PicReader::getData(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"); }

/*********************************************************************
*   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 arr(y, x, 4);
for (int i = 0;i < y;i++) {
    for (int j = 0;j < x;j++) {
        for (int k = 0;k < 4;k++) {
            arr[i][j][k] = data[i * y * 4 + j * 4 + k];
        }
    }
}
_x = x, _y = y;
return arr;


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
```