字符画实验报告

国豪 06 班 2251079 隋建政

2022.6.23

1. 设计思路与功能描述

1.1 设计思路

1.1.1 Array 类

首先需要实现一个 Array 类以实现矩阵的功能。我的 Array 类中包含了构造、析构、运算符重载、取已知位置元素数据以及重新规定 shape 的函数。在这些函数中实现起来比较困难的是[ ]的重载以及 at 函数的应用，但二者大体思路是类似的，都是通过 shape 和相应传入的参数确定相应的 index 来确定数据的位置，其中[ ]的重载会更复杂一些，因为该运算符需要使用多次，因此每次使用都是返回一个子矩阵，一步步直到最后返回 1*1 的数据即可。

1.1.2 PicReader 类

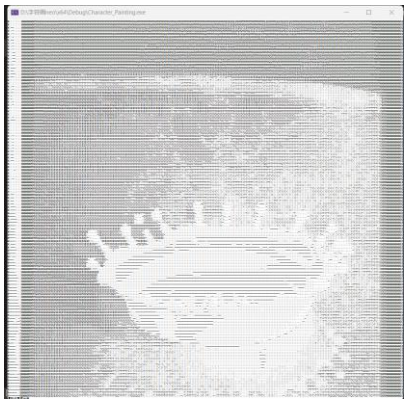
在该类中只修改 getData 函数的中间部分，我的思路是将画框范围以及一个 Array 类型的对象的地址传入，在函数中进行 Array 对象大小的初始化以及赋值的操作，同时把画框的宽度和高度都返回到主函数中。

1.2 功能实现

1.2.1 实现流程

首先打开图片后，利用 getData 将图片中每个像素点的 RGBA 值传入一个三维的 Array 对象中，将该 RGB 进行处理使其化为一个二维的 Array 图像，这就是图片的灰度值矩阵。然后利用不同的灰度，按灰度梯度给予灰度区间相应由深到浅的字符表示，再利用 PicReader 将字符画展示在屏幕上，每幅画之间我使用了 system（“pause”）进行暂留，在运行时按下任意键就可以观看下一张图片。

但在处理灰度值时还有些细节需要处理，如有些图片的对比度较低，单纯以灰度值生成字符画的效果并不好，因此我先是尝试了图片卷积的处理，但结果效果并不理想，轮廓线在卷积处理下被加重的太过夸张，该突出的重点反而又有所缺失。因此我又尝试了直方图均衡化的处理方法，简单来说就是根据不同像素点出现的概率动态的分配灰度值分布，该处理已经使得绝大多数图片的处理效果很好了，但对于 milkdrop 这张图片，处理效果还远远不够，因此我又对该图片进行了特判处理，加深两种色差之间的对比度，效果有了明显的改良。



特判处理前

特判处理后

2. 遇到的问题及解决方案

2.1 画面大小的问题

首先在 PicReader 构造时将字体大小设置为 6，若是不对画面的长宽做任何调整，最终输出的画面一定会出现断层，这就是大小超过限度的问题，因此我设置了步长，即每步长乘步长个像素点中只取其中一个，由于画面很大，因此这种取法在字符画中并不会表现出多大的缺陷。

其次若是对坐标不做任何处理，输出的字符并不是正方形而是长方形，因此图片呈现被压缩的状态，我首先尝试了将一个字符连续输出两遍的方法，效果可以说尚可，随后我又尝试了沿 y 轴每步长个取点，而延 x 轴每步长/2 就取一个点，这样正好满足字符高基本是长度二倍的条件，这种方法的最终呈现方法是明显好于第一种方法的。

2.2 画面正反的问题

在调试过程中，出现了画面反转和画面左右两部分重复的问题，在主函数中经过反复反转转换最终都得不到正确的图片。后来经过思考我意识到是在初始化 getData 函数中初始化 Array 类时就出现的问题，是因为画面的 xy 与我们惯常所理解的矩阵 ij 是不同的，因此将两项坐标反向，问题就解决了。

3. 心得体会

在完成该项目时我还是遇到了不少的问题，首先一点是在自己所写的 Array 类要实际应用时，就会发现很多在初期无法发现的问题，因此我想一个成功的类是要经历数据量的磨练的，只有通过数据的检验，才能发现类中很多的问题。

其次一点是惯常思维的问题，在制作过程中，我把图片的 xy 轴坐标直接对应矩阵中的 ij 元素，这种做法在处理正方形图片时还显现不出问题，但当真正处理长方形图片时，各式各样的错误便开始出现，我也为此浪费了很多时间，冥思苦想究竟是哪里的问题，所幸最终还是将问题排查了出来。

最后就是关于图片处理的一些小心得，在该程序的应用中我尝试了很多种不同的图像处理方法，例如卷积处理（虽然最终并没有应用）以及直方图均衡化，在处理 milkdrop 特判是我还是应用了 OSTU 算法的思想，但并不像 OSTU 算法中两种色差一刀切，而是增大两种颜色之间的色差以达到让图像对比度更高的效果。

源代码

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

class Array
{
```

```
public:
    template <typename... Args>
    Array(Args... args){
        auto num = int(sizeof...(args));
        int list[3] = { args... };
        int count = 1;
        for (int i = 0; i < num; i++) {
            count *= int(list[i]);
            shape[i] = int(list[i]);
        }
        data = new int[count * sizeof(int)];
        axisNum = num;
        nowAxis = 0;
        index = 0;
    }

    Array(Array& a)
    {
        axisNum = a.axisNum;
        index = 0;
        nowAxis = 0;
        int count = 1;
        for (int i = 0; i < axisNum; i++) {
            shape[i] = a.shape[i];
            count *= shape[i];
        }
        data = new int[count * sizeof(int)];
    }

    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[3] = { args... };

        index = 0;
        for (int i = 0; i < num; i++)
        {
            int size = 1;
            for (int j = i + 1; j < num; j++)
                size *= shape[j];
            index += size * int(list[i]);
        }
```

```
    return *this;
}

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... };

    for (int i = 0; i < axisNum; i++)
        shape[i] = 0;
    axisNum = num;
    for (int i = 0; i < axisNum; i++)
        shape[i] = int(list[i]);

}

int* get_content() { return data; }

void set(int value) { data[index] = value; }

Array& operator[](int in)
{
    // 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
    int size = 1;
    if (nowAxis == 0)
        index = 0;
    for (int i = nowAxis + 1; i < axisNum; i++) {
        size *= shape[i];
    }
    index += in * size;
    nowAxis++;
    nowAxis = nowAxis % axisNum;
    return *this;
}

Array& operator=(int data)
{
    this->data[index] = data;
    return *this;
}

Array& operator=(Array& a)
```

```
{
    int len = 1;
    for (int i = 0; i < a.axisNum; i++)
        len *= a.shape[i];
    for (int i = 0; i < len; i++)
        data[i] = a.data[i];
    return *this;
}

Array& operator+(Array& b)
{
    Array& sum = b;
    int len = 1;
    for (int i = 0; i < axisNum; i++)
        len *= shape[i];
    for (int i = 0; i < len; i++) {
        sum.data[i] = data[i] + b.data[i];
    }
    return sum;
}

Array& operator-(Array& b)
{
    Array& dif = b;
    int len = 1;
    for (int i = 0; i < axisNum; i++)
        len *= shape[i];
    for (int i = 0; i < len; i++) {
        dif.data[i] = data[i] - b.data[i];
    }
    return dif;
}

Array& operator*(Array& b)
{
    Array& pro = b;
    int len = 1;
    for (int i = 0; i < axisNum; i++)
        len *= shape[i];
    for (int i = 0; i < len; i++) {
        pro.data[i] = data[i] * b.data[i];
    }
    return pro;
}
```

```
    operator int() { return data[index]; }
    friend ostream& operator<<(ostream&, Array&);

    void set_shape(int x, int y)
    {
        shape[0] = x;
        shape[1] = y;
        shape[2] = 4;
        data = new int[x * y * 4 * sizeof(int)];
        axisNum = 3;
        nowAxis = 0;
        index = 0;
    }

    ~Array()
    {
        delete[]data;
    }
    int* data;
    int index;
    int shape[16];
    int axisNum;
    int nowAxis;
};

ostream& operator<<(ostream& out, Array& a)
{
    out << a.data[a.index];
    return out;
}

Array& operator+(Array& a, int add)
{
    Array& sum = a;
    int len = 1;
    for (int i = 0; i < a.axisNum; i++)
        len *= a.shape[i];
    for (int i = 0; i < len; i++) {
        sum.data[i] = a.data[i] + add;
    }
    return sum;
}
```

```
Array& operator-(Array& a, int drp)
{
    Array& dif = a;
    int len = 1;
    for (int i = 0; i < a.axisNum; i++)
        len *= a.shape[i];
    for (int i = 0; i < len; i++) {
        dif.data[i] = a.data[i] - drp;
    }
    return dif;
}

Array& operator*(Array& a, int k)
{
    Array& pro = a;
    int len = 1;
    for (int i = 0; i < a.axisNum; i++)
        len *= a.shape[i];
    for (int i = 0; i < len; i++) {
        pro.data[i] = a.data[i] * k;
    }
    return pro;
}

#define _CRT_SECURE_NO_WARNINGS
#include "Array.h"
#include "PicReader.h"
#include "FastPrinter.h"
#include <iostream>
#include <conio.h>
using namespace std;

void show(const char* name, int footstep)
{
    int x, y;
    PicReader imread;
    Array arr;
    imread.readPic(name);
    imread.getData(&x, &y, &arr);

    x = 2 * x / footstep;
    y = y / footstep;
    Array gray_pic(y, x);
```

```
    for (int i = 0; i < y; i++) {
        for (int j = 0; j < x; j++) {
            int arr_i = i * footstep;
            int arr_j = j * footstep / 2;
            gray_pic[i][j] = int(int(arr[arr_i][arr_j][0]) * 299 +
int(arr[arr_i][arr_j][1]) * 587 + int(arr[arr_i][arr_j][2]) * 114 + 500) / 1000;
        }
    }

    Array ave(y, x);
    int N = x * y;
    int count[256];
    double frq[256], frq_acu[256];
    for (int i = 0; i < 256; i++) {
        count[i] = 0;
        frq_acu[i] = 0;
    }
    for (int i = 0; i < y; i++)
        for (int j = 0; j < x; j++)
            count[int(gray_pic[i][j])]++;
    for (int i = 0; i < 256; i++) {
        frq[i] = double(count[i]) / N;
    }
    for (int i = 0; i < 256; i++) {
        for (int j = 0; j <= i; j++)
            frq_acu[i] += frq[j];
    }
    for (int i = 0; i < y; i++)
        for (int j = 0; j < x; j++) {
            ave[i][j] = int(frq_acu[int(gray_pic[i][j])] * 255);
            if (name == "classic_picture\\milkdrop.jpg") {
                if (ave[i][j] < 200)
                    ave[i][j] = int(ave[i][j]) - 30;
                else if (ave[i][j] >= 200 && ave[i][j] < 235)
                    ave[i][j] = int(ave[i][j]) - 50;
                if (ave[i][j] < 0)
                    ave[i][j] = 0;
            }
        }


    char asciiStrength[] = { 'M', 'N', 'H', 'Q', '$', 'O', 'C', '?',
                        '7', '>', '!', ':', '-', ';', '.' };
    // 灰度数据
```

```
    char ch[2];
    COORD text;
    FastPrinter printer(x, y, 6);
    printer.cleanSrceen();
    for (int i = 0; i < y; i++)
        for (int j = 0; j < x; j++) {
            text.X = j;
            text.Y = i;
            if(name!="classic_picture\\compa.png")
                ch[0] = asciiStrength[ave[i][j] / 18];
            else
                ch[0] = asciiStrength[gray_pic[i][j] / 18];
            ch[1] = '\0';
            printer.setText(text, ch, 0, 0, 0, 255, 255, 255);
        }
    printer.draw(true);
    system("pause");
}

int main()
{
    show("classic_picture\\lena1.jpg", 1);
    show("classic_picture\\lena.jpg", 3);
    show("classic_picture\\baboon.jpg", 4);
    show("classic_picture\\cameraman.jpg", 2);
    show("classic_picture\\peppers.jpg", 4);
    show("classic_picture\\woman.jpg", 4);
    show("classic_picture\\airplane.jpg", 4);
    show("classic_picture\\barbara.jpg", 5);
    show("classic_picture\\goldhill.jpg", 5);
    show("classic_picture\\compa.png", 2);
    show("classic_picture\\milkdrop.jpg", 4);
}
```