同濟大學
TONGJI UNIVERSITY

高级语言程序设计大作业

装

订

线

目录

装 订 线

1.设计思路与功能描述

1.1 设计思路：

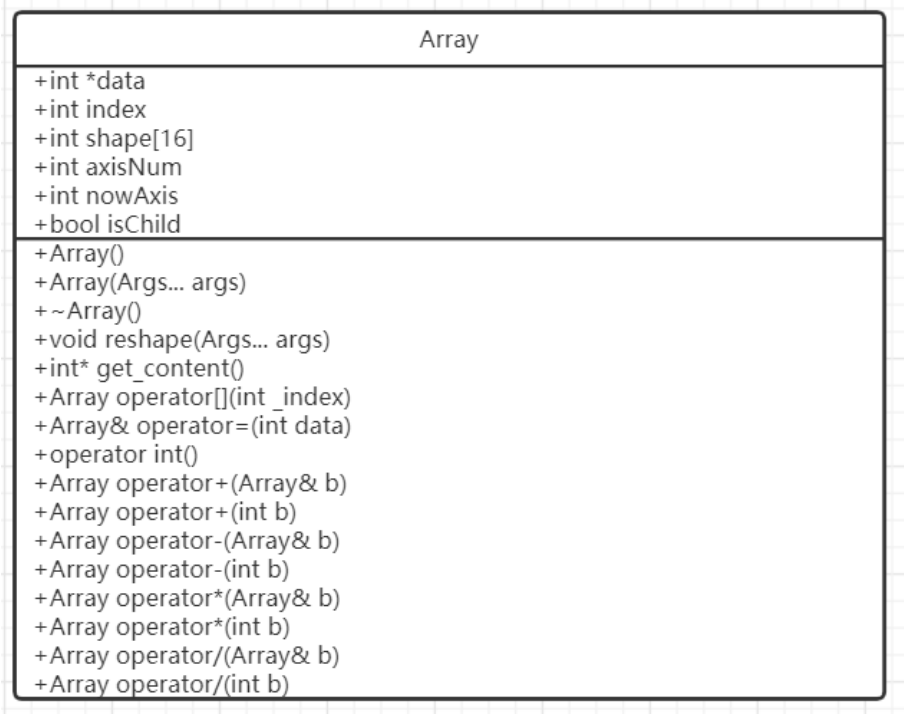
本次大作业共有三个主要任务，一是实现一个 Array 类，其难点在于通过重载[]实现多维数组；二是利用上述的 Array 类读取图片数据并处理，输出字符图片；三是实现一个 ASCII 视频播放器。

首先根据实现要求设计出 Array 类，根据所给框架设计类属性、类方法如下：

类属性：底层数据 data、data 的下标 index、存储每一维长度的 shape[16]、总维度数 axisNum、当前维度 nowAxis、判读是否为子矩阵的 isChild。

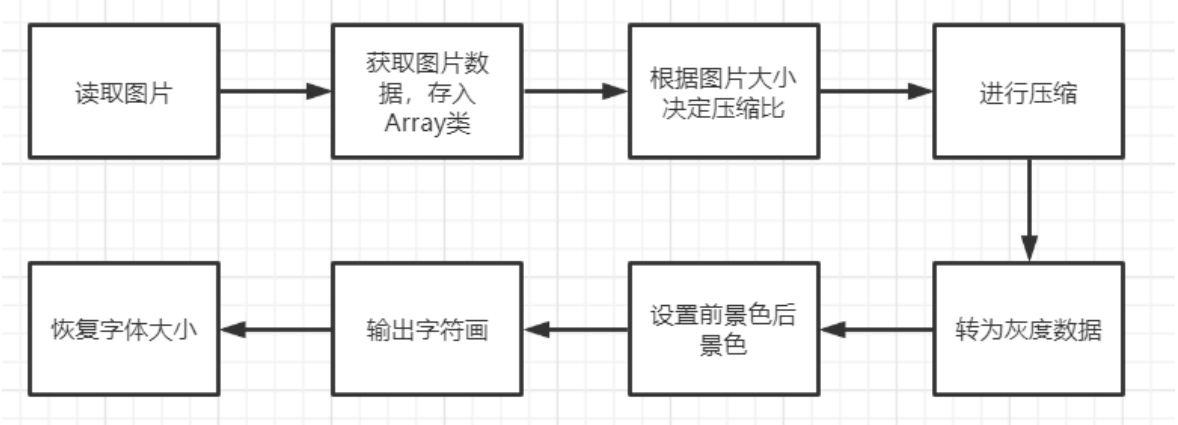
类方法：实现 reshape 操作的 reshape、获取底层一维数组的 get_content、重载[]实现多维数组、重载=实现数据赋值、operator int()返回当前指向的数据、重载+、-、*、/实现矩阵加减乘除。

类图如下：

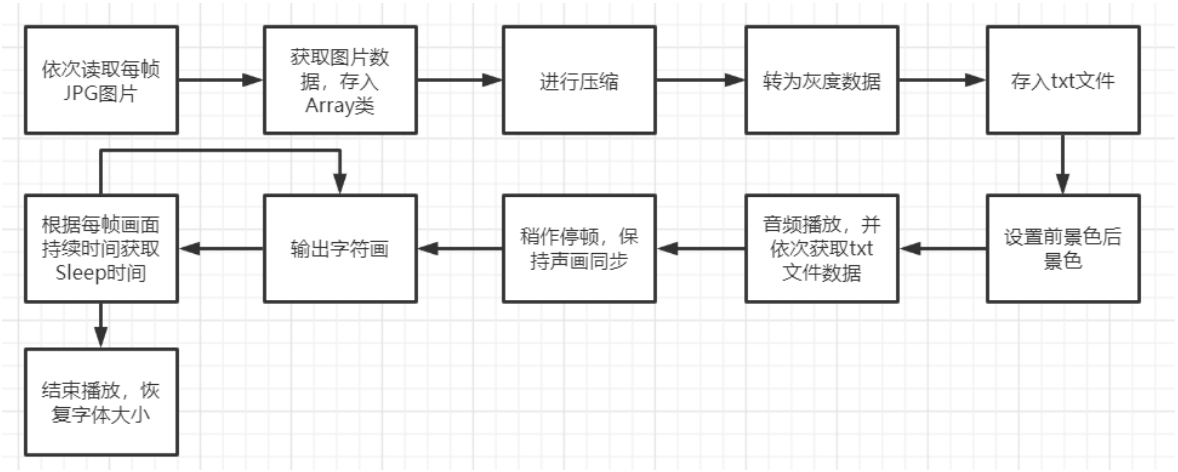


然后给出给出字符图片的生成过程以及字符视频播放器的实现过程：

字符图片生成过程如下：



字符视频播放器实现如下：



1.2 功能描述：

Array 类部分：

Array()：
    无参构造函数
~Array()：
    析构函数
Array(Args... args)：
    有参构造函数
void reshape(Args... args)：
    实现 reshape 操作
int* get_content()：
    获取底层一维数组
Array operator[](int _index)：
    重载[]实现多维数组
Array& operator=(int data)：
    重载=实现数据赋值
operator int()：
    返回当前指向的数据
Array operator+(Array& b)：
    矩阵加法
Array operator+(int b)：
    矩阵加法
Array operator-(Array& b)：
    矩阵减法
Array operator-(int b)：
    矩阵减法
Array operator*(Array& b)：
    矩阵点乘

Array operator*(int b)：
    矩阵点乘
Array operator/(Array& b)：
    矩阵点除
Array operator/(int b)：
    矩阵点除

顶层文件 main.cpp 部分：

void wait_for_enter()：
    等待输入回车键继续程序进程
void main_menu()：
    给出顶层的三大功能菜单：Array 类相关功能测试、字符图片输出、字符视频输出
int main():
    调用顶层的三大功能的相关函数

Array 类相关功能测试 array_func.cpp 部分：

void Array_menu()：
    给出相关测试菜单
void init():
    测试初始化
void print_ans();
    打印加减乘除结果
void matrixplus()：
    测试矩阵加法
void matrixsub()：
    测试矩阵减法：
void matrixdotp()：
    测试矩阵点乘
void matrixpointdiv()：
    测试矩阵点除
void martixreshape():
    测试矩阵 reshape
void getCstyle()：
    获取 C 风格元素操作
void Array_func()：
    从主函数进入，调用上述功能的相关函数

实现字符图片、字符视频输出的 pic_video.cpp 部分：

void get_video()：
    实现读取 txt 文件，并转为视频播放
void get_txt()：
    实现读取 JPG 文件，并转为 txt 文件

void Video_output()：

调用上述两函数，实现视频播放

void Pic_output()：

实现字符图片输出

补充的 PicReader.h 部分：

void getData(Array& DATA, UINT& _x, UINT& _y)：

补充该函数，实现读出来的图片数据存至 Array 内并 reshape 成(长 x 宽 x 通道)的三维矩阵进行保存

2.在实验过程中遇到的问题及解决方法

2.1 关于 Array 类：

因为 a[i]、a[i][j]、a[i][j][k]等等都是使用一样的 data，这些临时对象调用析构函数会造成同段空间多次删除，而且也会丢失原数据，如何解决？

加入一个变量 isChild，在析构时判断当前对象是否为子矩阵，是子矩阵则不删除内存，不是则删除，就可以避免这个问题。

2.2 关于字符图片：

2.2.1 如何解决如下的图片出现错位的情况？

是输出窗口高度过大，导致 RandCF 无法正常工作。所以需要通过压缩图片来缩减行数，解决错位问题。

2.2.2 如何解决图片比例错误？

OJ 建议可以尝试用两个字符表示一个像素；但同时结合 2.2.1 问题，通过使用一个字符代表共 4 行 2 列的 8 个像素的方式进行压缩，来保持图像比例正确。

2.2.370*70 图片太小，压缩后图片显示效果不明显，如何解决？

进行分情况处理，图片大小小于 200*200 的不进行压缩，大于 200*200 的采用一个字符代表共 4 行 2 列的 8 个像素的方式进行压缩。其实，更合理的解决方法是实现一个响应式压缩，根据图片长宽来选择成梯度的压缩比。

2.3 关于字符视频：

2.3.1 输出字符视频时内存占用飙升，如何解决？

原因是之前取数据时为防止析构函数删除数据，将 Array 对象设为了子矩阵，后来没有调整回来。只需将对象设为非子矩阵，及时进行动态内存释放即可。

2.3.2 如何解决图片转化速度太慢，导致视频播放卡顿问题？

原因是图片数据处理会消耗较长时间，因此先将图片数据处理后放入 txt 文件，然后依次读取每一个 txt 文件，并取出数据进行视频播放。

2.3.3 如何确定每帧图片播放时间？

首先记录下开始时间，然后根据当前时间和第 i 帧结束时的时间，去判断程序应该 Sleep 的时间。同时如果出现 Sleep 时间小于 0 时，取 Sleep 时间为一个较小值，即让接下来的几帧提前播放，使得后面 Sleep 时间相对延长，避免卡顿。

3. 心得体会

3.1 程序和算法设计的体会：

1）编程学习中，要注意底层实现逻辑，理解底层的工作原理，而不是一味的使用已经封装的库，这样才能真正提高编程水平。

2）对于 RGB 图转灰度图有各类不同的算法，除了 OJ 提供的算法，还有将除法转换为移位操作的算法，以及 Adobe RGB 算法等，应灵活掌握各类算法，积极做算法扩展，进行比较，选择最优。

3）程序设计要注重完备性、鲁棒性。对于程序要实现的功能，尽量写完整，写全面。尤其对动态内存的使用和回收，应注意调用析构函数的时间，内存回收的时间。

4）对于视频播放，主要是要注意及时进行内存释放，以及保持音画同步和按帧播放。

5）OJ 提到不允许使用 vector 进行矩阵实现，因此有一个思路是自己实现一个简单的 vector，这需要去了解 STL 源码。如果有时间也应当了解以下 vector、list、set、map 等 STL 容器的底层实现，对于理解栈、队列、红黑树等数据结构也有所帮助。

3.2 编程过程的体会：

1）一定要注意编程细节，比如内存的释放时间，压缩 4*2 矩阵块的选取等，一般程序出现问题，往往都是在细节上有漏洞。

2）程序中出现 bug 时，一定要注意 Debug。通过断点、单步调试、变量监控等快速锁定 bug，而且一般 bug 出现的位置自己也大概知道一个范围，所以往往选择自己感觉有问题的地方进行调试会很快找到 bug。

3）程序设计过程中要注重算法的学习，要积极寻找各类不同的算法，进行实践比较。

4）搜索引擎是自学过程中的好老师。在遇到自己不了解的部分时，可以去 cppreference、微软 c++文档、csdn、stackoverflow 等地方寻找解决方法。

4. 源代码：

Array.h:

```
#include<string.h>
class Array {
public:
    Array()//构造函数
    {
```

```
    index = 0;
    memset(shape, 0, sizeof(shape));
    axisNum = 0;
    nowAxis = 0;
    ischild = true;
    data = nullptr;
}
~Array()//析构函数
{
    if (!ischild)
    {
        delete data;
        data = nullptr;
    }
}
template<typename ...Args>
Array(Args... args) //构造函数
{
    // 获取参数包大小并转换为数组
    auto num = sizeof...(args);
    int list[100] = { args... };

    ischild = false;
    axisNum = num;
    nowAxis = 0;
    index = 0;
    //为每一维度的长度赋值
    memset(shape, 0, sizeof(shape));
    for (int i = 0; i <= axisNum; i++)
        shape[i] = 1;

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

    for (int i = 1; i < axisNum; i++)
    {
        shape[i] = shape[i - 1] / list[i - 1];
    }
    data = new int[shape[0]];
}
template<typename ...Args>
void reshape(Args... args)
{
    // 获取参数包大小并转换为数组
    auto num = sizeof...(args);
    int list[100] = { args... };
```

装 订 线

```
    axisNum = num;
    nowAxis = 0;
    index = 0;
    //调整每一维度长度
    memset(shape, 0, sizeof(shape));
    for (int i = 0; i <= axisNum; i++)
        shape[i] = 1;

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

    for (int i = 1; i < axisNum; i++)
    {
        shape[i] = shape[i - 1] / list[i - 1];
    }
}
int* get_content()
{
    return data;
}
Array operator[](int _index)//重载[]
{
    // 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 child;
    child.ischild = true;
    child.axisNum = axisNum - 1;
    child.nowAxis = nowAxis + 1;
    memcpy(child.shape, shape, 16);
    child.index = child.shape[child.nowAxis] * _index;

    if (child.axisNum == 0)
        child.data = data;
    else
        child.data = data + child.index;

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

```
}
Array operator+(Array& b)//矩阵加法
{
    Array c;
    c.index = 0;
    c.axisNum = axisNum;
    c.nowAxis = nowAxis;
    c.ischild = true;
    for (int i = 0; i <= axisNum; i++)
    {
        c.shape[i] = shape[i];
    }
    c.data = new int[c.shape[0]];
    for (int i = 0; i < c.shape[0]; i++)
    {
        c.data[i] = data[i] + b.data[i];
    }
    return c;
}
Array operator+(int b)//矩阵加法
{
    Array c;
    c.index = 0;
    c.axisNum = axisNum;
    c.nowAxis = nowAxis;
    c.ischild = true;
    for (int i = 0; i <= axisNum; i++)
    {
        c.shape[i] = shape[i];
    }
    c.data = new int[c.shape[0]];
    for (int i = 0; i < c.shape[0]; i++)
    {
        c.data[i] = data[i] + b;
    }
    return c;
}
Array operator-(Array& b)//矩阵减法
{
    Array c;
    c.index = 0;
    c.axisNum = axisNum;
    c.nowAxis = nowAxis;
    c.ischild = true;
    for (int i = 0; i <= axisNum; i++)
    {
```

装 订 线

```
            c.shape[i] = shape[i];
        }
        c.data = new int[c.shape[0]];
        for (int i = 0; i < c.shape[0]; i++)
        {
            c.data[i] = data[i] - b.data[i];
        }
        return c;
    }
    Array operator-(int b)//矩阵减法
    {
        Array c;
        c.index = 0;
        c.axisNum = axisNum;
        c.nowAxis = nowAxis;
        c.ischild = true;
        for (int i = 0; i <= axisNum; i++)
        {
            c.shape[i] = shape[i];
        }
        c.data = new int[c.shape[0]];
        for (int i = 0; i < c.shape[0]; i++)
        {
            c.data[i] = data[i] - b;
        }
        return c;
    }
    Array operator*(Array& b)//矩阵点乘
    {
        Array c;
        c.index = 0;
        c.axisNum = axisNum;
        c.nowAxis = nowAxis;
        c.ischild = true;
        for (int i = 0; i <= axisNum; i++)
        {
            c.shape[i] = shape[i];
        }
        c.data = new int[c.shape[0]];
        for (int i = 0; i < c.shape[0]; i++)
        {
            c.data[i] = data[i] * b.data[i];
        }
        return c;
    }
    Array operator*(int b)//矩阵点乘
```

装 订 线

```
{
    Array c;
    c.index = 0;
    c.axisNum = axisNum;
    c.nowAxis = nowAxis;
    c.ischild = true;
    for (int i = 0; i <= axisNum; i++)
    {
        c.shape[i] = shape[i];
    }
    c.data = new int[c.shape[0]];
    for (int i = 0; i < c.shape[0]; i++)
    {
        c.data[i] = data[i] * b;
    }
    return c;
}
Array operator/(Array& b)//矩阵点除
{
    Array c;
    c.index = 0;
    c.axisNum = axisNum;
    c.nowAxis = nowAxis;
    c.ischild = true;
    for (int i = 0; i <= axisNum; i++)
    {
        c.shape[i] = shape[i];
    }
    c.data = new int[c.shape[0]];
    for (int i = 0; i < c.shape[0]; i++)
    {
        c.data[i] = data[i] / b.data[i];
    }
    return c;
}
Array operator/(int b)//矩阵点除
{
    Array c;
    c.index = 0;
    c.axisNum = axisNum;
    c.nowAxis = nowAxis;
    c.ischild = true;
    for (int i = 0; i <= axisNum; i++)
    {
        c.shape[i] = shape[i];
    }
```

装 订 线

```
        c.data = new int[c.shape[0]];
        for (int i = 0; i < c.shape[0]; i++)
        {
            c.data[i] = data[i] / b;
        }
        return c;
    }

    int* data; //底层一维数组
    int index; //data 的下标
    int shape[16];//存储每一维的长度
    int axisNum; //总维数
    int nowAxis; //当前维数
    bool ischild; //判断是否是子矩阵决定是否析构
};
```

装

all.h：

```
#pragma once
#include <iostream>
#include<stdio.h>
#include<fstream>
#include<string>
#include<ctime>
#include<conio.h>
void Array_func();
void Pic_output();
void Video_output();
void wait_for_enter();
```

订

线

main.cpp：

```
#include"all.h"
using namespace std;
void wait_for_enter()//等待输入
{
    cout << endl << "按回车键继续";
    while (_getch() != '\r');
    cout << endl << endl;
}
void main_menu()//菜单
{
    cout << "*******************************" << endl;
    cout << "*     1 Array 类相关功能测试  *" << endl;
    cout << "*     2 字符图片输出          *" << endl;
    cout << "*     3 字符视频输出          *" << endl;
    cout << "*     0 退出系统              *" << endl;
    cout << "*******************************" << endl;
```

```
    cout << "选择菜单项<0~3>:" << endl;
    return;
}
int main()
{
    cout << "当程序画面暂停时，请按回车键以继续\n 图片和视频内容可根据注释信息更换
\n";
    // 定义相关变量
    char choice;
    char ch;
    while (true) //注意该循环退出的条件
    {
        wait_for_enter();
        system("cls"); //清屏函数
        main_menu(); //调用菜单显示函数
        cout << "按要求输入菜单选择项" << endl;
        cin >> choice;// 按要求输入菜单选择项 choice
        if (choice == '0') //选择退出
        {
            cout << "\n 确定退出吗?（Y or y）" << endl;
            cin >> ch;
            if (ch == 'y' || ch == 'Y')
                break;
            else
                continue;
        }
        switch (choice)
        {
        case '1':Array_func(); break;
        case '2':Pic_output(); break;
        case '3':Video_output(); break;
        default:
            cout << "\n 输入错误，请重新输入" << endl;
            wait_for_enter();
        }
        cout << endl;
    }
    return 0;
}
```

array_func.cpp：

```
#include"all.h"
#include"Array.h"
using namespace std;
Array a(2, 2), b(2, 2), c(2, 2), d(2, 2);//全局变量，方便测试矩阵加减乘除运算
void Array_menu()//菜单
```

```
{
    cout << "************************************************************" << e
ndl;
    cout << " *  1 初始化多维数组      2 矩阵加法      3 矩阵减
法      *" << endl;
    cout << " *  4 矩阵点乘            5 矩阵点除      6 reshape 操
作    *" << endl;
    cout << " *  7 获取 C 风格元素       0 退出系
统                    *" << endl;
    cout << "************************************************************" << e
ndl;
    cout << "选择菜单项<0~7>:" << endl;
    return;
}
void init()//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
{
    Array e(3, 3, 3); //3x3x3
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            for (int k = 0; k < 3; k++)
            {
                e[i][j][k] = i + j + k;
            }
    cout << "初始化后底层数据：\n";
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            for (int k = 0; k < 3; k++)
            {
                cout << e[i][j][k] << " ";
            }
    cout << endl;
}
void print_ans()//打印结果，用于矩阵加减乘除
{
    cout << "c:\n";
    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            cout << c[i][j] << " ";
        }
        cout << endl;
    }
    cout << "d:\n";
    for (int i = 0; i < 2; i++)
    {
```

```
        for (int j = 0; j < 2; j++)
        {
            cout << d[i][j] << " ";
        }
        cout << endl;
    }
}
void matrixplus()//矩阵加法
{
    if (a.shape[0] == b.shape[0] && a.shape[0] == c.shape[0] && a.shape[0] =
= d.shape[0])//相同矩阵才能做加法
    {
        c = a + b;
        d = a + 2;
    }
    print_ans();//打印结果
}
void matrixsub()//矩阵减法
{
    if (a.shape[0] == b.shape[0] && a.shape[0] == c.shape[0] && a.shape[0] =
= d.shape[0])
    {
        c = a - b;
        d = a - 2;
    }
    print_ans();
}
void matrixdotp()//矩阵点乘
{
    if (a.shape[0] == b.shape[0] && a.shape[0] == c.shape[0] && a.shape[0] =
= d.shape[0])
    {
        c = a * b;
        d = a * 2;
    }
    print_ans();
}
void matrixpointdiv()//矩阵点除,因为底层数据为 int 型，所以最后除法结果不是浮点数
{
    if (a.shape[0] == b.shape[0] && a.shape[0] == c.shape[0] && a.shape[0] =
= d.shape[0])
    {
        c = a / b;
        d = a / 2;
    }
    print_ans();
```

```
}
void martixreshape()//reshape 操作
{
    Array f(16);
    for (int i = 0; i < 16; i++)
        f[i] = i;
    f.reshape(4, 4);
    cout << "reshape 结果：\n";
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            cout << f[i][j] << " ";
        }
        cout << endl;
    }
}
void getCstyle()//获取 C 风格元数据操作
{
    Array g(4);
    for (int i = 0; i < 4; i++)
        g[i] = i;
    int* h = g.get_content();//数据类型取决于你的实现
    cout << "获取的 C 风格元素：\n";
    for (int i = 0; i < 4; i++)
        cout << h[i] << ' ';
}
void Array_func()
{
    //为矩阵 a,b 赋值，用于矩阵加减乘除
    int cnt = 1;
    for (int i = 0; i < 2; i++)
        for (int j = 0; j < 2; j++)
        {
            a[i][j] = cnt;
            b[i][j] = cnt;
            cnt++;
        }

    char choice;
    char ch;
    while (true) //注意该循环退出的条件
    {
        wait_for_enter();
        system("cls"); //清屏函数
        Array_menu(); //调用菜单显示函数
```

```
        cout << "按要求输入菜单选择项" << endl;
        cin >> choice;// 按要求输入菜单选择项 choice
        if (choice == '0') //选择退出
        {
            cout << "\n 确定退出吗?(Y or y)" << endl;
            cin >> ch;
            if (ch == 'y' || ch == 'Y')
                break;
            else
                continue;
        }
        switch (choice)
        {
        case '1':init(); break;
        case '2':matrixplus(); break;
        case '3':matrixsub(); break;
        case '4':matrixdotp(); break;
        case '5':matrixpointdiv(); break;
        case '6':martixreshape(); break;
        case '7':getCstyle(); break;
        default:
            cout << "\n 输入错误，请重新输入" << endl;
            wait_for_enter();
        }
        cout << endl;
    }
    //使得 a,b,c,d 可以析构
    a.ischild = false;
    b.ischild = false;
    c.ischild = false;
    d.ischild = false;
    return;
}
```

pic_video.cpp：

```
#pragma comment(lib, "winmm.lib")
#include"all.h"
#include"PicReader.h"
#include"FastPrinter.h"
using namespace std;
BYTE mydata[512][512][4];//把 Array 类的数据经处理后转入 mydata，避免 Array 类构造、析构频繁，拖慢程序
char asciistrength[] = { 'M','N','H','Q','$','O','C','?','7','>','!',':','-',';','.' };//将灰度分为 15 级
void get_video()
{
```

```
    clock_t CLOCKS_PER_1000SEC = ((clock_t)1);//用于时间数据转换
    const int SIZE = 160 * 45;//固定大小，，如果换别的视频，需要改变
    //const int SIZE = 320 * 90;
    BYTE* frontColorBuffer = new BYTE[SIZE * 3];
    BYTE* backColorBuffer = new BYTE[SIZE * 3];
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            frontColorBuffer[i * 3 + j] = 0;
            backColorBuffer[i * 3 + j] = 255;
        }
    }
    FastPrinter printer(160, 45, 5);
    //FastPrinter printer(320, 90, 5);
    PlaySoundA("resource1\\1.wav", NULL, SND_FILENAME | SND_ASYNC);//音频播
放
    //PlaySoundA("resource2\\2.wav", NULL, SND_FILENAME | SND_ASYNC);
    clock_t start = clock();//开始时间
    clock_t now;
    for (int z = 1; z <= 661; z++)//注意换视频要更换 661 为 501
    {
        if (z == 1)
            Sleep(700);
        string c = "resource1\\\\file\\\\";
        //string c = "resource2\\\\file\\\\";
        c += to_string(z);
        c += ".txt";
        ifstream fin(c, ios::binary);
        char* dataBuffer = new char[SIZE];
        fin.read(dataBuffer, SIZE);//文件读取
        printer.cleanSrceen();
        printer.setData(dataBuffer, frontColorBuffer, backColorBuffer);
        printer.draw(true);
        now = clock();//获取当前时间
        //确定 Sleep 时间
        double sleeptime = double(start +double(700)+ 1000 * double(z) / 24
- now / CLOCKS_PER_1000SEC);
        if (sleeptime < 0)
            sleeptime = 10;
        Sleep((DWORD)sleeptime);
        delete[] dataBuffer;
    }
    delete[] frontColorBuffer;
    delete[] backColorBuffer;
    getchar();
```

```
    //恢复字体大小
    CONSOLE_FONT_INFOEX cfi;
    cfi.cbSize = sizeof(cfi);
    GetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi);
    cfi.dwFontSize.X = 0;
    cfi.dwFontSize.Y = 16;
    SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi);
}
void get_txt()
{
    const int SIZE = 160 * 45;//大小固定，如果换第二个视频，需要用下面注释的
    //const int SIZE = 320 * 90;
    PicReader imread;
    UINT x, y;
    BYTE* frontColorBuffer = new BYTE[SIZE * 3];
    BYTE* backColorBuffer = new BYTE[SIZE * 3];
    //JPG 转 TXT
    for (int z = 1; z <= 661; z++)//注意换视频要更换 661 为 501
    {
        Array data;
        //形成字符串，作为文件路径
        string a = "resource1\\\\pic\\\\(";
        string c = "resource1\\\\file\\\\";
        //string a = "resource2\\\\pic\\\\(";
        //string c = "resource2\\\\file\\\\";
        a += to_string(z);
        a += ").jpg";
        char b[25];
        memset(b, 0, sizeof(b));
        for (int i = 0; i < a.size(); i++)
        {
            b[i] = a[i];
        }
        c += to_string(z);
        c += ".txt";

        imread.readPic(b);
        imread.getData(data, x, y);
        data.ischild = false;//使得 data 可析构
        //压缩
        for (UINT k = 0; k < y; k += 4)
        {
            for (UINT l = 0; l < x; l += 2)
            {
                int sumr = 0;
                int sumg = 0;
```

```
                int sumb = 0;
                int suma = 0;
                for (int i = 0; i < 4; i++)
                {
                    for (int j = 0; j < 2; j++)
                    {
                        sumr += data[k + i][l + j][0];
                        sumg += data[k + i][l + j][1];
                        sumb += data[k + i][l + j][2];
                        suma += data[k + i][l + j][3];
                    }
                }
                mydata[k / 4][l / 2][0] = sumr / 8;
                mydata[k / 4][l / 2][1] = sumg / 8;
                mydata[k / 4][l / 2][2] = sumb / 8;
                mydata[k / 4][l / 2][3] = suma / 8;
            }
        }
        //转灰度
        char* dataBuffer = new char[SIZE];
        char asciistrength[] = { 'M','N','H','Q','$','O','C','?','7','>','!'
,':','-',';','.' };
        for (UINT i = 0; i < y / 4; i++)
        {
            for (UINT j = 0; j < x / 2; j++)
            {
                BYTE graydata = (mydata[i][j][0] * 299 + mydata[i][j][1] * 5
87 + mydata[i][j][2] * 114 + 500) / 1000;
                BYTE asciiindex = graydata / 18;
                dataBuffer[(i * x / 2 + j)] = asciistrength[asciiindex];
            }
        }

        //txt 写入文件
        ofstream fout(c, ios::binary);
        if (!fout)
        {
            cerr << "Can not open the output file!" << endl;
            exit(-1);
        }
        fout.write((char*)&dataBuffer[0], SIZE);
        fout.close();
        delete[] dataBuffer;
        dataBuffer = nullptr;
    }
    delete[] frontColorBuffer;
```

```
    delete[] backColorBuffer;
    cout << "转 txt 结束\n";
    getchar();
}
void Video_output()
{
    cout << "JPG 文件已经转成 txt 文件并放在 resource1 和 resource2 下 file 文件夹里，所以可以直接输入 2 得到动画\n";
    cout << "如果要换视频，请注意更换文件路径，图片大小等信息，相关代码已用注释形式放在相关位置\n";
    cout << "输入 1，将 JPG 文件转换为字符数组，并存入 txt 文件，可作为验证 JPG 转 txt 的方式\n";
    cout << "输入 2，读取 txt 文件，得到字符画视频\n";
    int n;
    cin >> n;
    if (n == 1)
    {
        get_txt();
    }
    else if (n == 2)
    {
        get_video();
    }
    else
    {
        cout << "输入错误，返回主菜单\n";
        getchar();
        return;
    }
}
void Pic_output()
{
    PicReader imread;

    //图片
    imread.readPic("classic_picture\\airplane.jpg"); //512*512
    //imread.readPic("classic_picture\\baboon.jpg"); //512*512
    //imread.readPic("classic_picture\\barbara.jpg"); //720*580
    //imread.readPic("classic_picture\\cameraman.jpg");//256*256
    //imread.readPic("classic_picture\\compa.png"); //385*184
    //imread.readPic("classic_picture\\goldhill.jpg"); //720*576
    //imread.readPic("classic_picture\\lena.jpg"); //400*400
    //imread.readPic("classic_picture\\lena1.jpg"); //70*70
    //imread.readPic("classic_picture\\milkdrop.jpg");//512*512
    //imread.readPic("classic_picture\\peppers.jpg"); //512*512
    //imread.readPic("classic_picture\\woman.jpg"); //512*512
```

装 订 线

```
UINT x, y;
Array data;
imread.getData(data, x, y);//获取 data、x、y
data.ischild = false;//使得 data 可以析构
int SIZEX, SIZEY, compressX, compressY;//x 长度，y 长度，x 压缩量，y 压缩量
if (x <= 200 && y <= 200)//小于 200*200 不压缩
{
    compressX = compressY = 1;
}
else//按 x 方向取 2，y 方向取 4 的 2*4 小块，作为压缩数据的一个小块进行压缩
{
    compressX = 2;
    compressY = 4;
}
SIZEX = x / compressX;
SIZEY = y / compressY;
int SIZE = SIZEX * SIZEY;//SIZE 为压缩后总数据量
int x1 = SIZEX * compressX;//x 方向向下取偶数
int y1 = SIZEY * compressY;//y 方向向下取 4 的倍数
//压缩
for (int k = 0; k < y1; k += compressY)
{
    for (int l = 0; l < x1; l += compressX)
    {
        int sumr = 0;
        int sumg = 0;
        int sumb = 0;
        int suma = 0;
        for (int i = 0; i < compressY; i++)
        {
            for (int j = 0; j < compressX; j++)
            {
                sumr += data[k + i][l + j][0];
                sumg += data[k + i][l + j][1];
                sumb += data[k + i][l + j][2];
                suma += data[k + i][l + j][3];
            }
        }
        mydata[k / compressY][l / compressX][0] = sumr / 8;
        mydata[k / compressY][l / compressX][1] = sumg / 8;
        mydata[k / compressY][l / compressX][2] = sumb / 8;
        mydata[k / compressY][l / compressX][3] = suma / 8;
    }
}
//如果不压缩，那 x 方向数据乘 2，避免图片变形
```

```
    if (compressX == 1 && compressY == 1)
        SIZE *= 2;
    char* dataBuffer = new char[SIZE];
    BYTE* frontColorBuffer = new BYTE[(size_t)SIZE * 3];
    BYTE* backColorBuffer = new BYTE[(size_t)SIZE * 3];
    //转为灰度数据
    for (int i = 0; i < SIZEY; i++)
    {
        for (int j = 0; j < SIZEX; j++)
        {
            BYTE graydata = (mydata[i][j][0] * 299 + mydata[i][j][1] * 587 +
mydata[i][j][2] * 114 + 500) / 1000;
            BYTE asciiindex = graydata / 18;
            if (compressX == 1 && compressY == 1)
            {
                int pos = 2 * (i * SIZEX + j);
                if (pos < SIZE)
                {
                    dataBuffer[pos] = asciistrength[asciiindex];
                    dataBuffer[pos + 1] = asciistrength[asciiindex];
                }
            }
            else
            {
                int pos = i * SIZEX + j;
                if (pos < SIZE)
                    dataBuffer[pos] = asciistrength[asciiindex];
            }
        }
    }
    //前景色、后景色赋值
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            frontColorBuffer[i * 3 + j] = 0;
            backColorBuffer[i * 3 + j] = 255;
        }
    }
    //输出字符画
    if (compressX == 1 && compressY == 1)
    {
        SIZEX *= 2;
        FastPrinter printer(SIZEX, SIZEY, 5);
        printer.setData(dataBuffer, frontColorBuffer, backColorBuffer);
        printer.draw(true);
```

```
        getchar();
        getchar();
        //恢复字体大小
        CONSOLE_FONT_INFOEX cfi;
        cfi.cbSize = sizeof(cfi);
        GetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi
);
        cfi.dwFontSize.X = 0;
        cfi.dwFontSize.Y = 16;
        SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi
);
    }
    else
    {
        FastPrinter printer(SIZEX, SIZEY, 5);
        printer.setData(dataBuffer, frontColorBuffer, backColorBuffer);
        printer.draw(true);
        getchar();
        getchar();
        //恢复字体大小
        CONSOLE_FONT_INFOEX cfi;
        cfi.cbSize = sizeof(cfi);
        GetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi
);
        cfi.dwFontSize.X = 0;
        cfi.dwFontSize.Y = 16;
        SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi
);
    }

    delete[] dataBuffer;
    delete[] frontColorBuffer;
    delete[] backColorBuffer;

    return;
}
```

PicReader.h 补充部分：

```
void /*TO-DO: 可能你需要修改返回类
型 END*/ PicReader::getData(Array& DATA, 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, da
ta);
    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 mydata(4*x*y);
    mydata.ischild = true;//防止 mydata 在这里被析构
    for (UINT i = 0; i < 4 * x * y; i++)
    {
        mydata[i] = data[i];
    }
    mydata.reshape(y, x, 4);
    DATA = mydata;
    _x = x;
    _y = y;
    delete[] data;

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

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