大作业4分享

1954020 刘章丁
1950574 王朝宇

具体内容摘要

图像转ascii大作业分享

1

Array类的构建

2

图片转ascii

3

连续图片转视频

1

Array
类

1、实现一个Array类，应至少实现以下功能：

（1）至少能初始化三个维度的数组（通过重载[]）

（2）具备矩阵基础操作功能：矩阵加减法、矩阵点除

点乘

（3）实现reshape，即将一个一维数组reshape为二维

或三维

（4）获取指针

右图是第一版，只能初始化三个维度的
数组。可以作为试验。

```
class Array {
public:
    int m, n, l;
    unsigned char ***p;
public:
    void first_set();    //初始化m*n*1矩阵,动态内存申请
    Array(int, int, int);//构造函数
    Array(int, int);
    Array(int);
    Array();
    ~Array();
    unsigned char** operator[](int);
    friend Array operator+(Array, Array);
    friend Array operator-(Array, Array);
    friend Array operator*(Array, Array);
    friend Array operator/(Array, Array);
    Array reshape(int, int, int);
    Array reshape(int, int);
    Array reshape(int);
    Array reshape(BYTE*, int, int);
    unsigned char* get_content();
    void display();
};
```

- 第一次返回二维指针
- m*n*l大小
- Array(int, int, int) 构造函数(3个)
- 对应写3个reshape函数

```
void Array::first_set()
{
    p = new unsigned char**[m];
    for (int i = 0; i < m; i++) {
        p[i] = new unsigned char*[n];
        for (int j = 0; j < n; j++)
            p[i][j] = new unsigned char[l];
    }
}
```

- 但是无法实现任意维度。a[i]=i//因为是返回指针 错
- 而且写起来复杂，下面介绍array框架实现

```
Array a(16);
for (int i = 0; i < 16; i++) a[i] = i;
```

```
unsigned char** Array::operator[](int i)//....
{
    return p[i];
}
```

可变模版参数

```
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... };
}
```

- C++11新增的最强大的特性之一，它对参数进行了高度泛化
- 它能表示0到任意个数、任意类型的参数 //可以实现 至少能初始化三个维度的数组
  - 可变参数模板和普通模板的语义是一样的，只是写法上稍有区别，声明可变参数模板时需要在typename或class后面带上省略号“...”。
  - 比如我们常常这样声明一个可变模版参数：template<typename...>或者template<class...>，一个典型的可变模版参数的定义是这样的：

```
template <class... T>
void f(T... args);
```

  - 省略号的作用有两个：
  - 1.声明一个参数包T... args，这个参数包中可以包含0到任意个模板参数；
  - 2.在模板定义的右边，可以将参数包展开成一个一个独立的参数。
  - 上面的参数args前面有省略号，所以它就是一个可变模版参数，我们把带省略号的参数称为“参数包”，它里面包含了0到N（N>=0）个模版参数。

- f()没有传入参数，所以参数包为空，输出的size为0，后面两次调用分别传入两个和三个参数，故输出的size分别为2和3。
- 展开可变模版参数函数的方法一般有两种：
- 一种是通过递归函数来展开参数包，
- 一种是通过逗号表达式来展开参数包。

```
template <class... T>
void f(T... args)
{
    cout << sizeof...(args) << endl; //打印变参的个数
}

f();        //0
f(1, 2);    //2
f(1, 2.5, "");    //3
```

递归函数方式展开参数包

- 需要提供一个参数包展开的函数和一个递归终止函数
- 会输出每一个参数，直到为空时输出empty。展开参数包的函数有两个，一个是递归函数，另外一个是递归终止函数，参数包Args...在展开的过程中递归调用自己，每调用一次参数包中的参数就会少一个，直到所有的参数都展开为止，当没有参数时，则调用非模板函数print终止递归过程。
- 递归调用的过程如右

  分别输出1 2 3 4

```
#include <iostream>
using namespace std;
//递归终止函数
void print()
{
    cout << "empty" << endl;
}
//展开函数
template <class T, class ...Args>
void print(T head, Args... rest)
{
    cout << "parameter " << head << endl;
    print(rest...);
}


int main(void)
{
    print(1, 2, 3, 4);
    return 0;
}
```

```
//print(1, 2, 3, 4);
//print(2, 3, 4);
//print(3, 4);
//print(4);
//print();
```

- 递归函数展开参数包是一种标准做法，也比较好理解，但也有一个缺点,就是必须要一个重载的递归终止函数，即必须要有一个同名的终止函数来终止递归，这样可能会感觉稍有不便。

逗号表达式展开参数包

- expand函数中的逗号表达式：(printarg(args), 0)，也是按照这个执行顺序，先执行printarg(args)，再得到逗号表达式的结果0。
- 同时还用到了C++11的另外一个特性——初始化列表，通过初始化列表来初始化一个变长数组, {(printarg(args), 0)...}将会展开成((printarg(arg1),0), (printarg(arg2),0), (printarg(arg3),0), etc...)，
- 最终会创建一个元素值都为0的数组int arr[sizeof...(Args)]。
- 由于是逗号表达式，在创建数组的过程中会先执行逗号表达式前面的部分printarg(args)打印出参数，也就是说在构造int数组的过程中就将参数包展开了，这个数组的目的纯粹是为了在数组构造的过程展开参数包。

```
template <class T>
void printarg(T t)
{
    cout << t << endl;
}

template <class ...Args>
void expand(Args... args)
{
    int arr[] = { (printarg(args), 0)... };
}

expand(1, 2, 3, 4);
```

- **本质上都是一维度（可以进行reshape多维的原因），重载[]时改变一维数据的索引**
- 错误判断：参数个数一致；数组的大小必须大于0；用变量记录总长度当reshape16->[4][5]时报错等
- 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：

```
temp.index = temp.index * shape[i] + list[i];
```

- 比如array a(3,4) ,如右图
- 对 a[2][3]
- 1.a[2] index=2
- 2. a[2][3] index=2*4+3=11
- 3.a[2][3]在一维上索引是11

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

- reshape时判断前后规模是否一致，把维度写到shape数组里
- 矩阵加减法、点除点乘必须要同型矩阵（可以先判断是否同型），再对底层一维数组操作
- 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原数据。

```
int* get_content() {
    return data;
}
```

- 涉及动态内存申请，定义拷贝构造函数data new，datap记录原来的data。防止delete多次。

```
Array(const Array& temp) {//论const的重要性
```

```
class"Array": 没有可用的复制构造函数或复制构造函数声明为"explicit"
```

- 注意const

```
Array& operator=(int data) {
    this->data[index] = data;//重载=后可以赋值
    return *this;
}
operator int() {
    return data[index];// a[][]可以作为右值
}
```

- 其他

q

注意：这里复制构造函数作为参数传进去的那个对象前加个const.

- 如果在函数中不会改变引用类型参数的值，加不加const的效果是一样的。而且不加const，编译器也不会报错。但是为了整个程序的安全，还是加上const，防止对引用类型参数值的意外修改。
- 返回出来的对象是const的，而我这里传进复制构造函数的对象不是const的，所以出问题了

- 那么为什么要用引用呢？
- 【错误】个人第一反应：为了减少一次内存拷贝。
- 【正确】参数不是引用时Array（ temp）。在执行temp=Array(a);时，其实会调用拷贝构造函数。如果我们的拷贝构造函数的参数不是引用，会调用Array temp = a;，又因为temp之前没有被创建，所以又需要调用拷贝构造函数，故而又执行Array temp = a;，就这样永远的递归调用下去了。
- 所以，拷贝构造函数是必须要带引用类型的参数的，而且这也是编译器强制性要求的。

[]：返回对象而不是引用（多重[]）

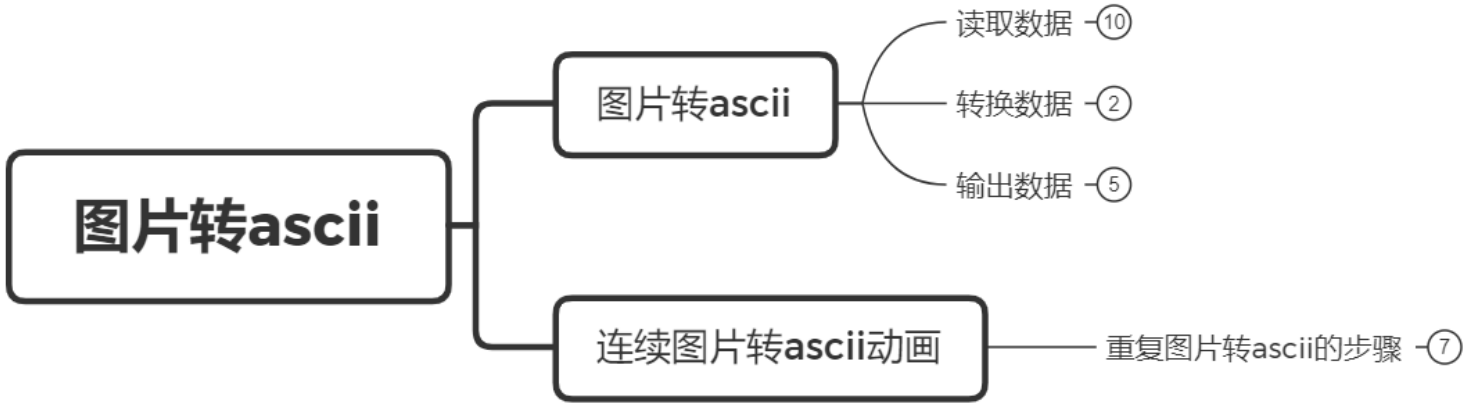
- 第一反应引用，因为这样data始终是最开始的指针
- 但如果是引用，为了进行错误判断，需要保留nowAxis的值
- 当[]使用正确的时候，nowAxis也会被保留下来，那么什么时候置零呢？
- 我能想到的是调用运算符的时候，因为只有这时候才能确定[]调用完了，这样就要求每个运算符内都要加上置零的操作，太麻烦。

- 另一方面，可能有一些不包含运算符的调用[]的情况，这样nowAxis不能置零，之后会出现问题

2

图片转ascii

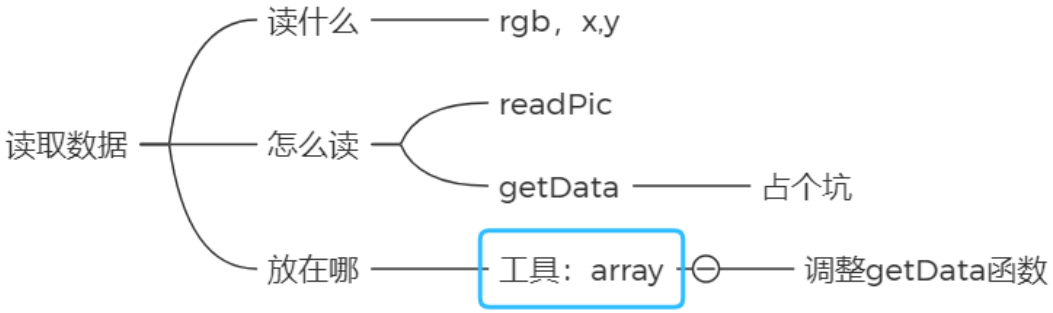
思路介绍



图片转ascii：流程很简单，读取、转换、输出

连续图片转ascii：重复上述过程，对函数进行微调

读取数据



读什么：rgb（转换），xy（初始化Array、FastPrinter）

怎么读：readPic，getData

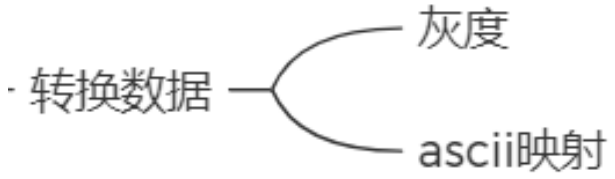
放在哪：Array，需要改造getData，调整参数和返回类型，（不改参数的原因：本来考虑的是不知道型号，但实际上可以调用参数个数为0的构造函数，此时不new，到了里面再初始化），定义局部对象，对其初始化，将其作为返回值初始化外面的Array。

都做完了，不过还有个问题——数据会不会太多，到输出的时候再说。

```
Array /*TO-DO: 可能你需要修改返回类型 END*/ PicReader::getData(/*TO-DO: 这里可能需要修改传参 END*/) {
```

大作业3

```
void readLena( char* chr) {
    PicReader imread;
    BYTE* data = nullptr;
    UINT x, y;
    imread.readPic((LPCSTR) chr);
    imread.getData(data, x, y);
    printf("show lena, press enter to continue...");
    (void)getchar();
    imread.showPic(data, x, y);
    delete[] data;
    data = nullptr;
    printf("Press enter to continue...");
    (void)getchar();
```



转换数据

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

1，将灰度数据转换为ASCII字符

最简单的，你可以采用映射的形式，即每一个数据段采用一个ASCII进行表示，并且这能很好的解决ASCII字符不足的问题，给出参考如下。

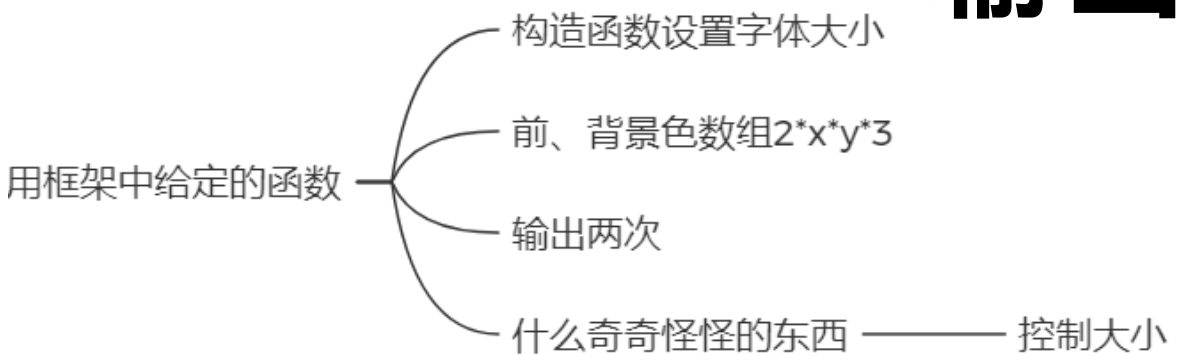
C++ Copy

```
//·将灰度分为15级，每一级由一个ascii字符表示，强度越大ascii字符内容越少(越白)
char·asciiStrength[]·=·{·'M','N','H','Q','$','O','C','?','7','>','!',':','-',';','.'·};

//·灰度数据
unsigned·char·grayData[]·=·{·154,80,97,15,214,32,68,90·};

for·(int·i·=·0;·i·<·8;·i++)·{
····//·255·/·18·=·14,故不会导致越界且能划分出强度等级
····unsigned·char·asciiIndex·=·grayData[i]·/·18;
····cout·<<·asciiStrength[asciiIndex];
}
```

Tips: 你可以通过加大等级、改善灰度映射算法或通过分析纹理选择合适字符来达到更好的效果。

Oj上都给了，拿过来用就好了

输出数据




```
FastPrinter(size_t, size_t);
FastPrinter(size_t, size_t, WORD);
```

```
for (unsigned i = 0; i < 2 * x * y * 3; i++) {
    frontColorBuffer[i] = 0;
    backColorBuffer[i] = 255;
}
```

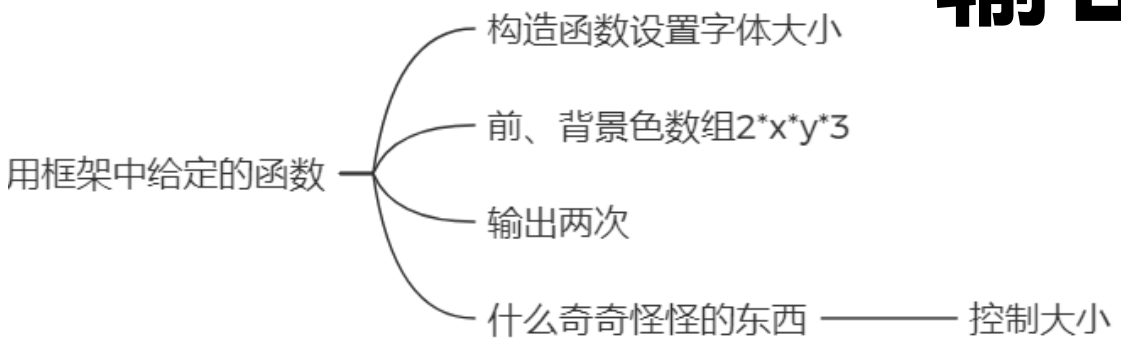
框架中给的内容：FastPrinter、setData。
一些细节：
调用FastPrinter类的构造函数时要初始化控制台大小和字体大小。（如果不？）（fontsize 6）
前景色、后景色数组是rgb不是灰度
由于字体横纵比的关系，字符要重复两次

大功告成？什么奇奇怪怪的东西

```
void setData(const char*, const BYTE*, const BYTE*);
void setData(const char*, const BYTE*, const BYTE*, SMALL_RECT);
void setRect(SMALL_RECT, const char, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void fillRect(SMALL_RECT, const char, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void setText(COORD, const char*, const size_t, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void setText(COORD, const char*, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void setText(COORD, const char*);
```


选择E:\chrome下载\Z_4_Resources\Z_4_Resources\x64\Debug\Z_4_Resources.exe

输出数据




```
FastPrinter(size_t, size_t);
FastPrinter(size_t, size_t, WORD);
```

```
for (unsigned i = 0; i < 2 * x * y * 3; i++) {
    frontColorBuffer[i] = 0;
    backColorBuffer[i] = 255;
}
```

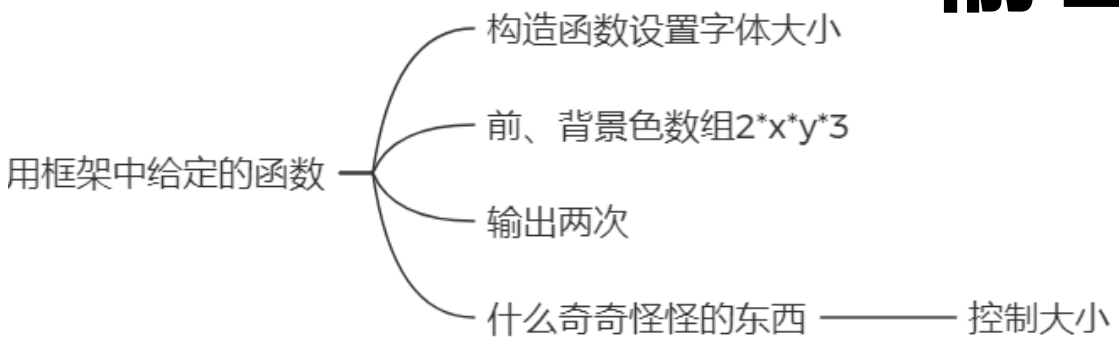
输出用框架中给的函数：setData。
一些细节：
在此之前，要用构造函数初始化控制台大小和字体大小。（6）
前景色、后景色数组是rgb不是灰度
由于字体横纵比的关系，字符要重复两次

大功告成？——什么奇奇怪怪的东西

```
void setData(const char*, const BYTE*, const BYTE*);
void setData(const char*, const BYTE*, const BYTE*, SMALL_RECT);
void setRect(SMALL_RECT, const char, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void fillRect(SMALL_RECT, const char, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void setText(COORD, const char*, const size_t, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void setText(COORD, const char*, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE, const BYTE);
void setText(COORD, const char*);
```


E:\chrome下载\Z_4_Resources\Z_4_Resources\x64\Debug\Z_4_Resources.exe

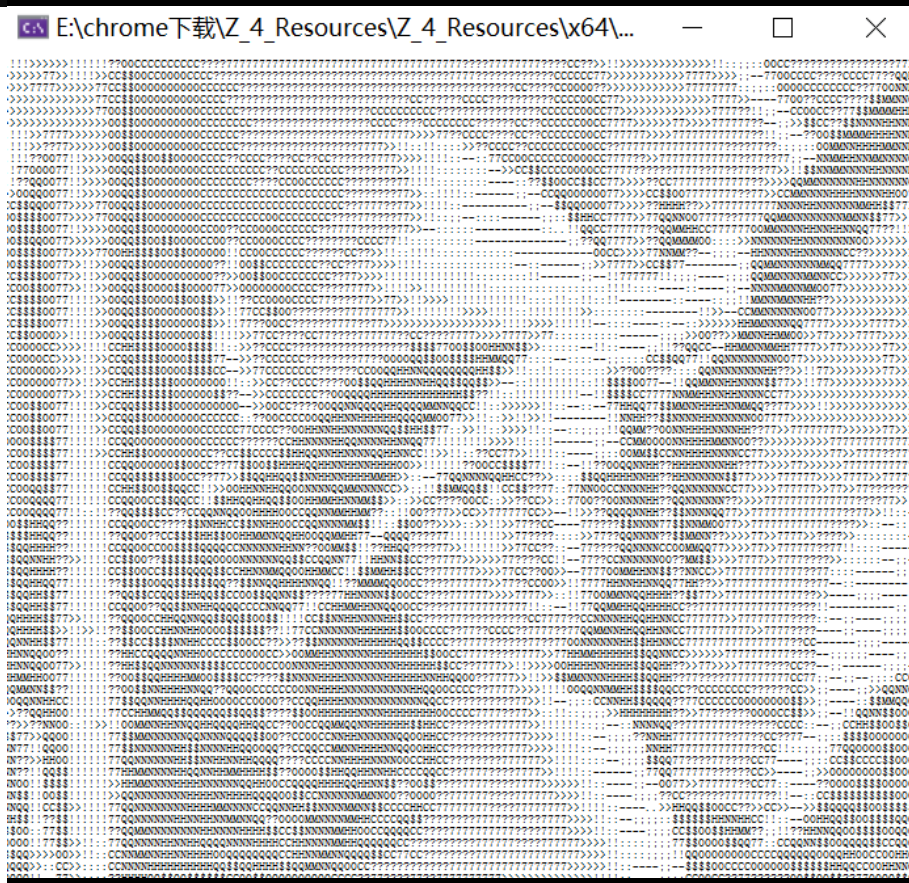
输出数据



框架中给的函数：
输出用框架提供的函数setData。
在此之前，要用构造函数初始化控制台大小和字体大小
前景色、后景色数组是rgb不是灰度
由于字体横纵比的关系，字符要重复两次

大功告成？——什么奇奇怪怪的东西
图片太大，需要控制大小

怎么控制？
在数据读进来、转换成灰度后：如果比屏幕的size大，缩放
d=（int（实际长or宽 /最大长or宽）+1）倍。新Array型号x/d,y/d，
其灰度值是原来图形中d*d个灰度值的平均值。控制台大小2x/d,y/d

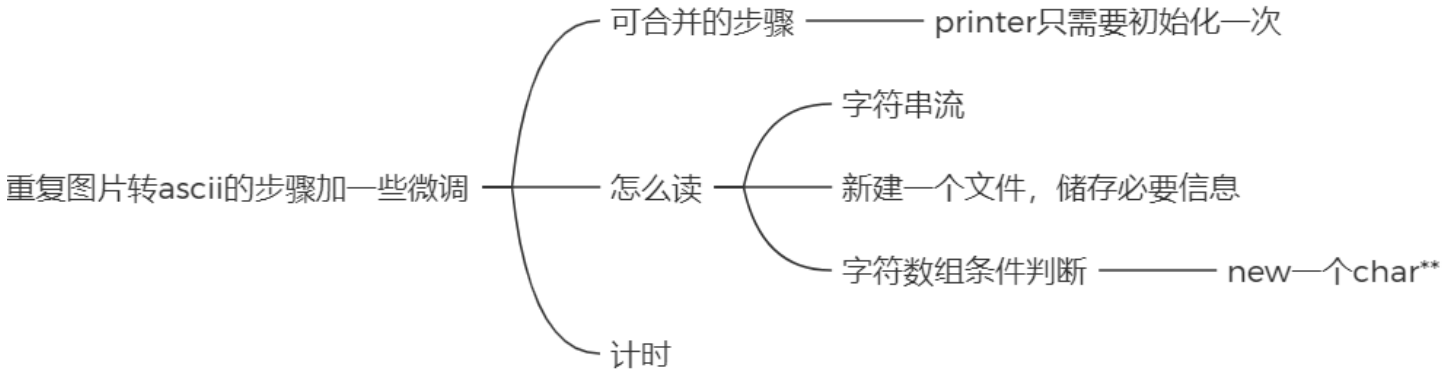



```
//窗口限制在合适的大小
int d = 0;
const int windowsize = 150;
if (x / windowsize >= y / windowsize)
    d = x / windowsize+1 ;
else
    d = y / windowsize+1 ;//d是较大的倍数
```

3

连续图像转ascii视频

连续图像转ascii视频



转视频时，图片转ascii的有些步骤做一次就好了（各种对象变量的初始化，如FastPrinter, Array， PicReader对象的定义，前景色、后景色数组的赋值），函数中删除相应步骤

完成前置工作后，就是不断的读图片，重复图片转ascii的步骤。
那么怎样读图片呢?
char*会有一些条件判断；字符串流很方便，根据整数位数找好输入的位置就好（注意对尾0覆盖的位置重新赋值）
为了音画同步需要计时，参照上周同学的分享就好

（先txt再动画还是直接边读图片边动画）（逐步调试没考虑初始化的问题）

连续图像转ascii视频

4

花里胡哨

Python 是世界上最好的语言!!!--wwsdl

python简单实现（浏览思路）

- 彩色图像，转换为灰度图像
- 灰度图像有256级 • 每个级别用一个字符表示
- 输出字符集即可

定义字符集

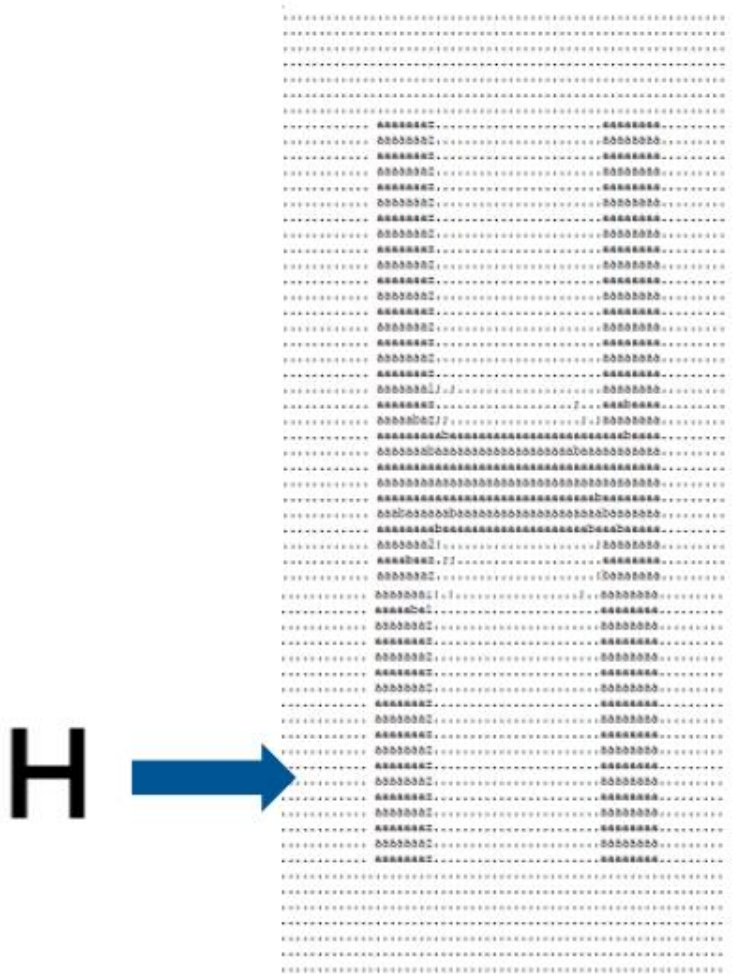
传递一个像素的r,g,b , 转化为灰度值

灰度/256, 算出其对应的位置

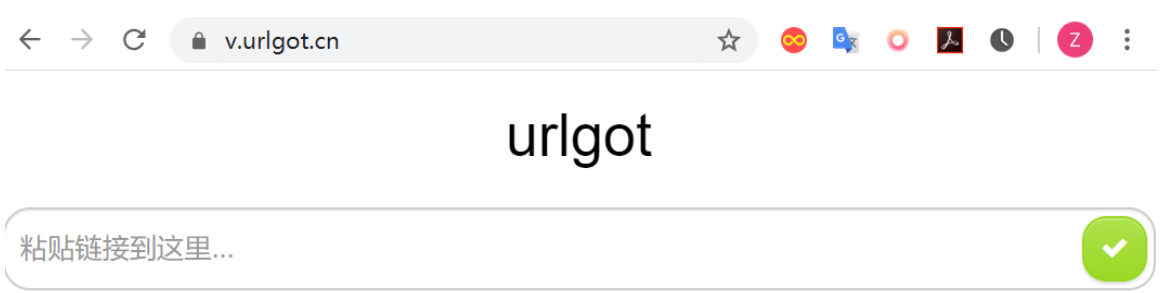
位置*字符集长度，取整n，从字符集中取出第n字符

遍历图像，生成图像每行对应的字符串，输出

- from PIL import Image
- charList='abcdefghijklmnopqrstuvwxyz1234567890!#$%^\
- &*ABCDEFGHIJKLMNOPQRSTUVWXYZ<>~`":;.'　　//我们可以优化字符集
- def getChar(rgb): # 传递一个像素的rgb值
- r,g,b=rgb # 解包
- gray=int(0.2126*r+0.7152*g+0.0722*b) # 转灰度
- unit=gray/256 #灰度在256级中的位置，因为字符集没有
- 256个字符
- return charList[int(unit*len(charList))]　　//灰度分类
- im = Image.open(r"大写H.jpg")
- im.show()
- w,h=im.size　　// 传递图片高和宽
- txt=''
- for i in range(h):
- for j in range(w):
- txt +=getChar(im.getpixel((j,i)))
- txt +='\n'
- fo=open(r"char_pic.txt",'w')
- fo.write(txt)
- fo.close()
- **********代码结束。　可以看到图像边缘化的问题，可进行模糊处理，加噪点



一个网站 https://v.urlgot.cn/



①复制b站的视频链接，打开下边这个网站，按 ctrl + v 把链接粘贴到那个框中.

②点击要下载清晰度后边的蓝色按钮即可

感谢聆听

QIANYU PPT