



计算原理导论

Introduction to Computing Principles

张鹏

天津大学 智能与计算学部



关于老师



张鹏

天津大学，智能与计算学部

办公地点：天津大学北洋园校区55-B529

Email: pzhang@tju.edu.cn

电话：18202251892

主页：<http://cic.tju.edu.cn/faculty/zhangpeng>

- 天津大学智能与计算学部副教授，计算机学院副院长，入选天津大学北洋学者-青年骨干教师计划，微软亚洲研究院铸星计划。目前担任中国中文信息学会青年工作委员会委员、中国人工智能学会青年工作委员会委员。发表论文40余篇，包括中国计算机学会（CCF）推荐A/B类会议论文（SIGIR, ACL, IJCAI, AAAI, WWW, CIKM, EMNLP）和期刊论文（TNNLS, TKDE, TIST, IP&M）。获得顶级学术会议 SIGIR 2017 Best Paper Award Honorable Mention, 欧洲信息检索会议 ECIR 2011 Best Poster Award。

课程简介



- 结合天津大学国际化趋势，作为《大学计算机基础》的教改课程，在教学内容、授课语言均做出改革。
- 参照美国斯坦福大学网上[公开课程CS101](#)所用素材，自主制定课件。
- 全英文授课，授课对象为非计算机专业大一新生。
- 通过应用实例和浅显的语言，理解计算原理。

课程目的



本课程从应用计算机的角度扩展学生的视野，使学生能在一个较高层次上认识和应用计算机，为学生进一步学习计算机技术奠定基础。



课程信息



Course Title	计算原理导论 (Introduction to Computing Principles)
Course Time	4-19周, 周一下午5-6节, 周三下午7-8节 (单周)
Class Location	上课: 周一 46A202, 周三46A202 (单周) 上机:
Course ID	建筑工程1班/2班 化工材料1班/2班/3班 机械工程1班/2班 求是英才班1班/2班
Teaching Assistants	骆克: keluo@tju.edu.cn 张晔华: yehua_zhang@tju.edu.cn
Office Hour and Location	
Course Homepage	待定
FTP Server	ftp://211.81.52.6

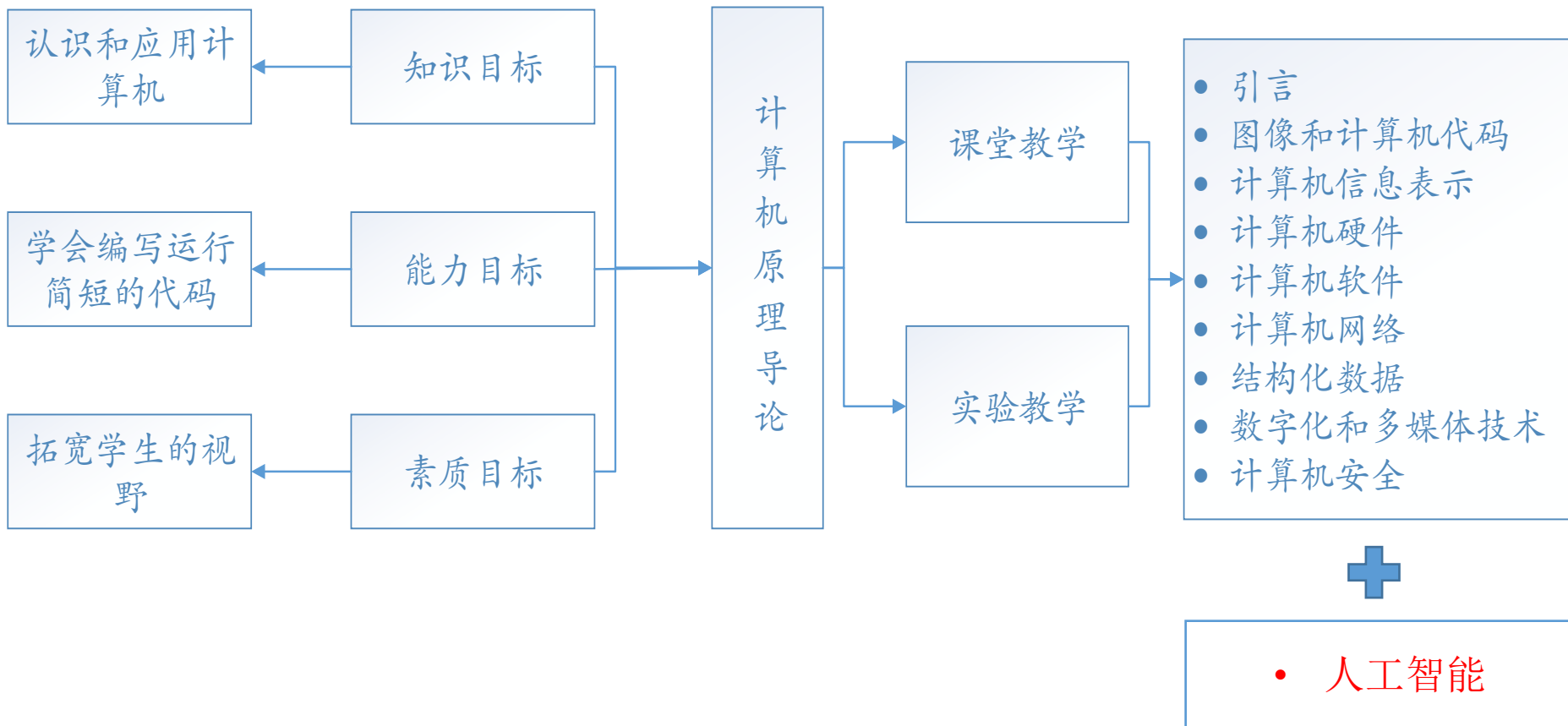
课程信息



2019-2020 学年第一学期日程安排表

周数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
日期	08.19-08.25	08.26-09.01	09.02-09.08	09.09-09.15	09.16-09.22	09.23-09.29	09.30-10.06	10.07-10.13	10.14-10.20	10.21-10.27	10.28-11.03	11.04-11.10	11.11-11.17	11.18-11.24	11.25-12.01	12.02-12.08	12.09-12.15	12.16-12.22	12.23-12.29
周一				上课	上课	上机	上机		上课	上机	上机	上课	上课	上机	上课	上课	上机	上机	上机
周二																			
周三					上课				上课		上机		上课		上课		上机		上机
周四																			
周五																			
周六								上课											考试
周日																			
备注																			

课程框架与章节



- Stanford Online Lagunita
 - <https://lagunita.stanford.edu/courses/Engineering/CS101/Summer2014/>
- The Stanford Website by Nick
 - <http://web.stanford.edu/class/cs101/>
- Fundamentals of Computer Science (4rd edition):
By Behrouz Forouzan, 2018

参考资料



Stanford University

Stanford
ONLINE
Lagunita

ABOUT

FIND COURSES

zhilei_liu ▾

MY COURSES

Help



Computer Science 101

Engineering - CS101-SP
Started - Jul 15, 2014



View Course

OUR RESEARCH COMMUNITY

Stanford University pursues the science of learning. Online learners are important participants in that pursuit. The information we gather from your engagement with our instructional offerings makes it possible for faculty, researchers, designers and engineers to continuously improve their work and, in that process, build learning science.

By registering as an online learner, you are also participating in research...

[Read Terms of Service and Privacy Policy.](#)

Stanford
ONLINE
Lagunita

Terms of Service
(Revised October 26, 2015)

Privacy Policy
(Revised October 26, 2015)

Honor Code

Copyright

Contact

Help

POWERED BY
OPENEDX

Stanford
University

SU Home

Maps & Directions

Search Stanford

Terms of Use

Copyright Complaints

© Stanford University, Stanford, California 94305

参考资料



Stanford University

Stanford ONLINE LEARNING Engineering: CS101-SP Computer Science 101 zhilei_liu

Courseware Course Info Discussion Wiki Progress

Help

- Week 1
 - Stanford CS101 Introduction
 - Code Writing
 - Code Writing Exercises Exercises
 - Code Variables
 - Code Variables Exercises Exercises
 - Digital Images**
 - Digital Image Exercises Exercises
 - Image Code
 - Image Code Exercises Exercises
 - Week 2
 - Week 3
 - Week 4
 - Week 5
 - Week 6

Introduction to Digital Images

- White light - broken up into pure colors, continuous
- Red, orange, yellow, green, blue, indigo, violet (ROY G BIV)
- Pure colors as a palette
- Run the experiment backwards to make white
- Funny thing about "indigo"

paper. So what this shows is that white light is not some indivisible pure thing. Instead, it can be separated into these pure constituent colors. And so this is the same thing that you would see with a rainbow. So the colors are actually in a continuous spectrum. But we Newton identified certain ones. You know gave them, gave them words. So it goes, this is the famous sequence where it goes red on the side here, and then orange, and yellow, and green. And finally blue, indigo and violet on the far side here. So. In the computer. I want to think of these pure colors as kind of a palette. And the scheme we're gonna use is

Download video Download transcript .srt

参考资料



Stanford University

Stanford ONLINE Lagunita Engineering: CS101-SP Computer Science 101 zhilei_liu

Courseware Course Info Discussion Wiki **Progress**

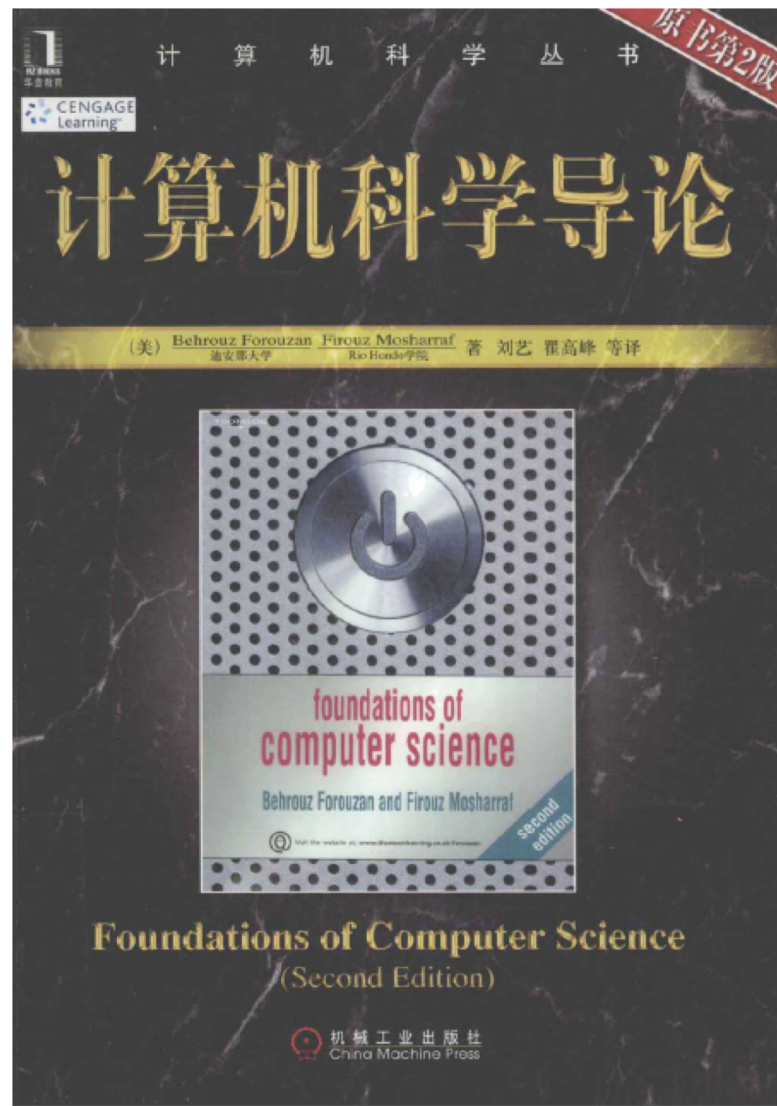
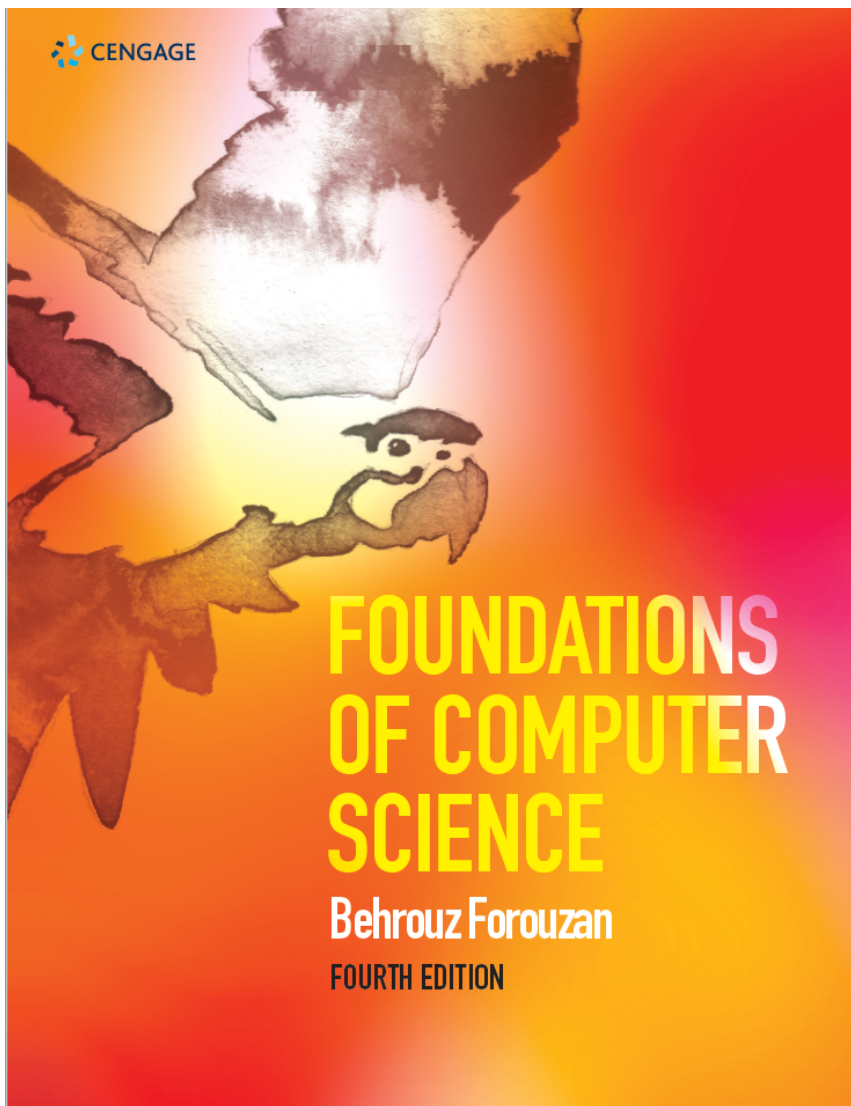
Course Progress for Student 'zhilei_liu' (zhileiliu@tju.edu.cn)

Note: Submitted assignments will appear as bars in the Progress chart. Mouse over a bar to see more information about the relevant assignment.

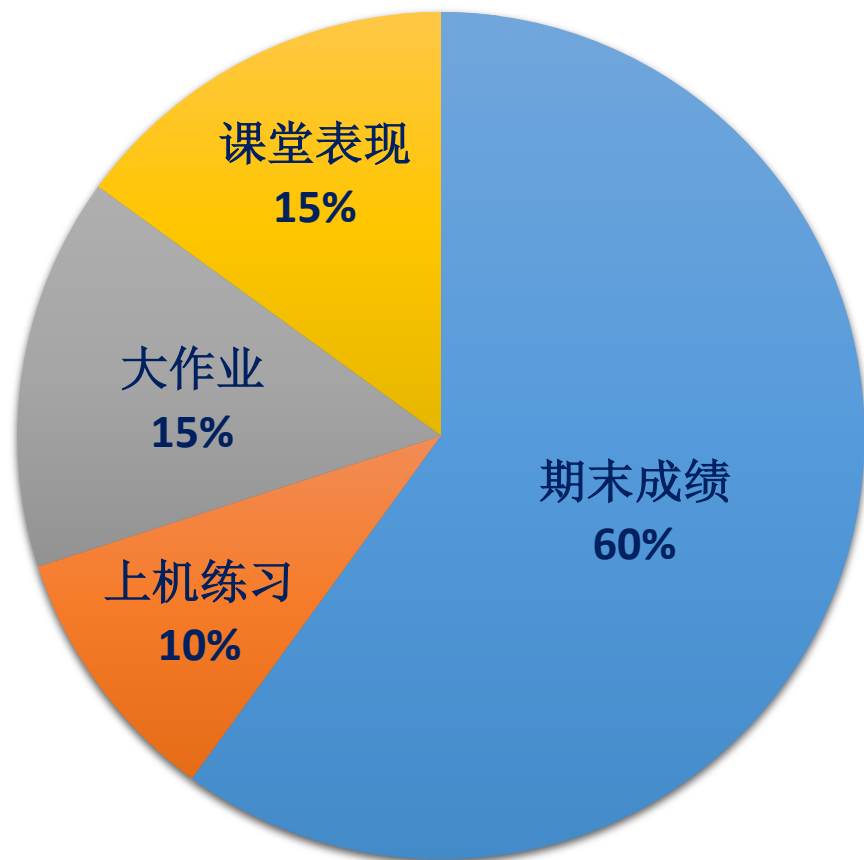
Assignment	Progress (%)
Exercises 01	100%
Exercises 02	100%
Exercises 03	100%
Exercises 04	100%
Exercises 05	0%
Exercises 06	0%
Exercises 07	0%
Exercises 08	0%
Exercises 09	0%
Exercises 10	0%
Exercises 11	0%
Exercises 12	0%
Exercises 13	0%
Exercises 14	0%
Exercises 15	0%
Exercises 16	0%
Exercises 17	0%
Exercises 18	0%
Exercises 19	0%
Exercises 20	0%
Exercises 21	0%
Exercises 22	0%
Exercises 23	0%
Exercises 24	0%
Exercises 25	0%
Exercises 26	0%
Exercises 27	0%
Exercises 28	0%
Exercises 29	0%
Exercises Avg	14%
Total	14%

Stanford ONLINE Lagunita Terms of Service (Revised October 26, 2015) Privacy Policy (Revised October 26, 2015) Honor Code Copyright Contact Help POWERED BY OPENEDX

参考资料



成绩组成



1. Exercises和Homework均交到指定FTP文件夹，标题格式：专业+班级+学号+姓名+练习/作业号，如：建工1班+2018*****+张三+ImageEx1
2. 2次翻转课堂
3. 3次大作业
4. 三次缺勤取消考试资格

Let's start to learn Stanford CS101: Introduction to Computing Principles



- Basic ideas of how computer works
- Not a magic box, but basics understandable
- Look inside, how they work
- Understand what they can do, cannot do
- No computer background required at all
- Be curious!



• Fundamental Equation of Computers

Computer = Powerful + Stupid

Powerful

- looking through masses of a data
- Billions of "operations" per second

Stupid

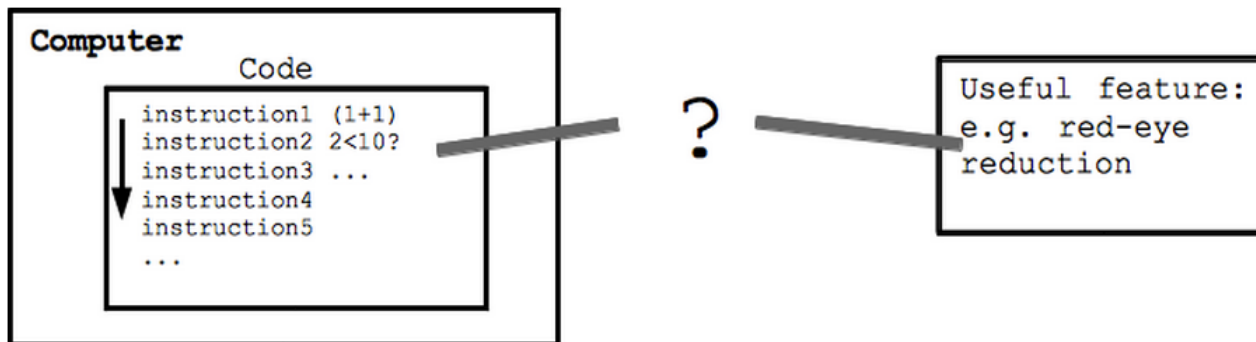
- Operations are simple and mechanical
- Lacks "insight" [[video](#)]

Will computers ever be smarter than humans?



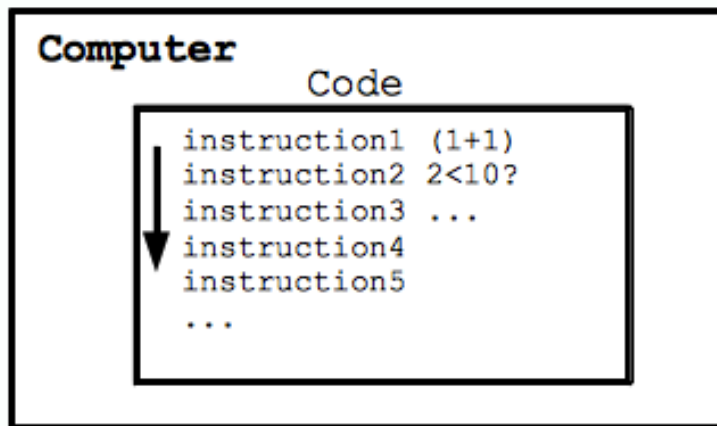
How Does a Computer Work?

- **Question:** if the computer is so mechanical, how are they so useful? What connects the two sides?
- Think of all the useful computer features (phone, camera)
 - --Email, instant messaging
 - --MP3 audio
 - --Red-eye reduction



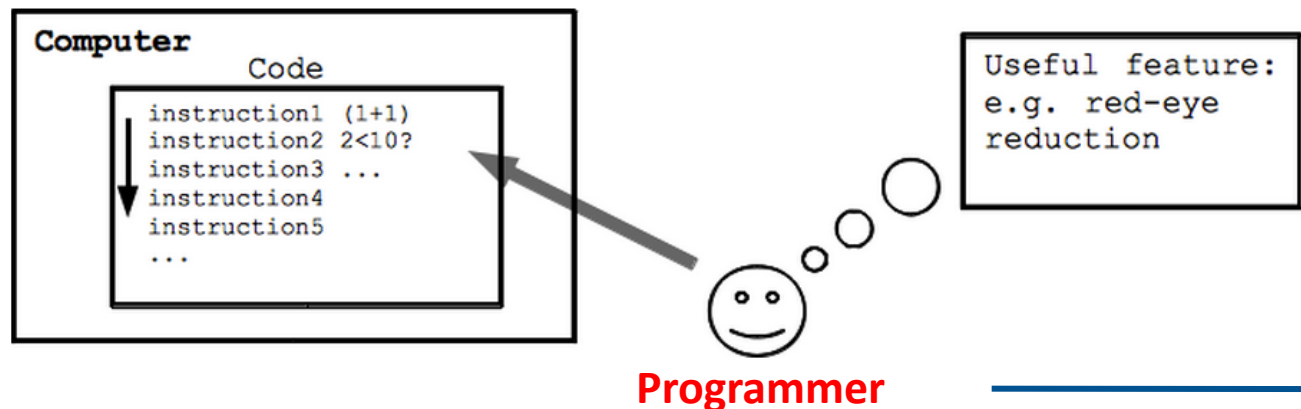
How Does a Computer Work?

- Computer is driven by "code" (more detail later)
- Code is made of simple, mechanical instructions
- Computers do not have insight



How Does a Computer Work?

- Human programmer thinks of a useful feature
 - Creativity, insight about problems, computers, ...
- Programmer thinks through the solution → **Algorithm**
- Breaking it down, directing the computer → **Programing**
- Combine the best features of both sides
 - insight + inexpensive/fast



- We'll start with some simple coding below
- First code examples are not flashy
- Code is like Lego bricks, eventually make great combinations
- But we have to start small
- In the near soon, we'll be doing special effects with images:



- Javascript is a high-level, dynamic, untyped, and interpreted programming language.
- JavaScript is the programming language of HTML and the Web.
- Programs are small ... not professional Javascript.

- **Note:** Java \neq Javascript





- Javascript is one of the three core technologies of World Wide Web content production.
- JavaScript is easy to learn.
- Our code phrases are small, but big enough to show the real issues, bugs of coding.



• First Code Example – Print

```
print(6);  
print(1, 2);
```

6
1 2

code1-1

Run

- **Run (运行)** executes each line once, running from top to bottom
- **print** is a **function (函数)** -- like a verb in the code
- Numbers within the parenthesis (...) are passed in to the print function
- Multiple values separated by **commas**

[Exercise](#)

Syntax(语法)



What is Syntax?

```
print("a");  
print(1, "b");  
print(2, "c", 3);
```

Run



Error:Unexpected token ILLEGAL



- **Syntax:** In computer science, the syntax of a computer language is **the set of rules** that defines the combinations of **symbols** that are considered to be a correctly structured document or fragment in that language.
- Code is not free form
 - Syntax is limited, computer can only understand the code in this form
- **Note:** "print" is not a normal part of Javascript, just for this class.



- A **string** is a sequence of letters written **within quotes** to be used as data within the code
 - e.g. "hello" ; "world "
 - Strings work with the print function, in addition to numbers
- Strings in the computer store **text**, such as urls or the text of paragraphs, etc.
- **Comment** begins with // and extends through the end of the line.
 - A way to write notes about the code, ignored by the computer.

```
// The line below prints one
number and one string
print(6, "hi");
print("hello", 2, "bye");
```

```
6 hi
hello 2 bye
```



Syntax Errors

- Very common error -- type in code, with slight syntax problem
- Professional programmers make that sort of "error" all the time
- Fortunately, very easy to fix ... don't worry about it
- Not a reflection of some author flaw
- Demo: a bunch of typical syntax errors + fixing them
- Beginners can be derailed by syntax step
- Do not be intimidated by these quick little errors



Syntax Error Examples

```
print("a");  
print(1, "b");  
print(2, "c", 3);
```

```
print("a");  
print(1, "b");  
print(2. "c". 3):
```

```
print("a");  
print(1, "b");  
print(2, "c", 3;
```

a

Error:print is not defined

Error:Unexpected token ILLEGAL

Error:missing) after argument list

Exercise

Variable (变量)

What is variable and what variable can do?

```
print (3)
```



3

```
x=3;
```

```
print (x)
```

3

Definition

A variable or scalar is a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity of information referred to as a value

A "variable" is like a box that holds a value

Code

```
x = 7;
```



x

Assignment

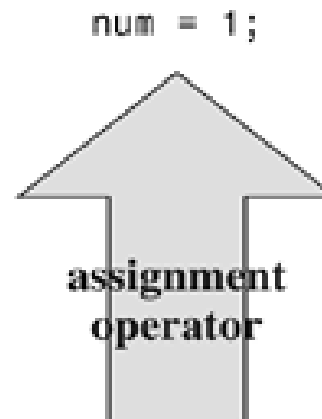
- $x = 7;$
- This particular example means "assign the value 7 to the variable x ".
- You can assign x a different value later, if you want; that is why x is termed a **variable**.
- Using $=$ in this way is called "variable assignment"

Other examples:

```
num = 1;
```

```
y = 2 ;
```

```
z = y+2;
```





Name Notice

- You should use **meaningful** names for variables (such as `sheep_count` instead of `x3` if your program counts sheep.). If the name doesn't suffice, use comments to explain what the variables represent. **Documenting a program in this manner is one of the basic techniques of good programming.**
- The characters at your disposal are **lowercase letters, uppercase letters, digits, and the underscore (`_`)**. **The first character must be a letter or an underscore.** The following are some examples:

Valid Names

`wiggles`

`cat2`

Invalid Names

`$Z]**`

`2cat`



Experiments

- try assigning (=) these values to x: 8, "hi"
- The name **x** can be anything we choose, so long as we are consistent
 - Try changing it to **xyz** throughout
- The Point: store a value once, use it on several lines, can save repetition
- **Not the same as = in algebra, that means two things are equal forever**
- = in code is simple -- just put a value in a box when that line runs

Gold Puzzle (谜图)



Here we have the Gold Puzzle image

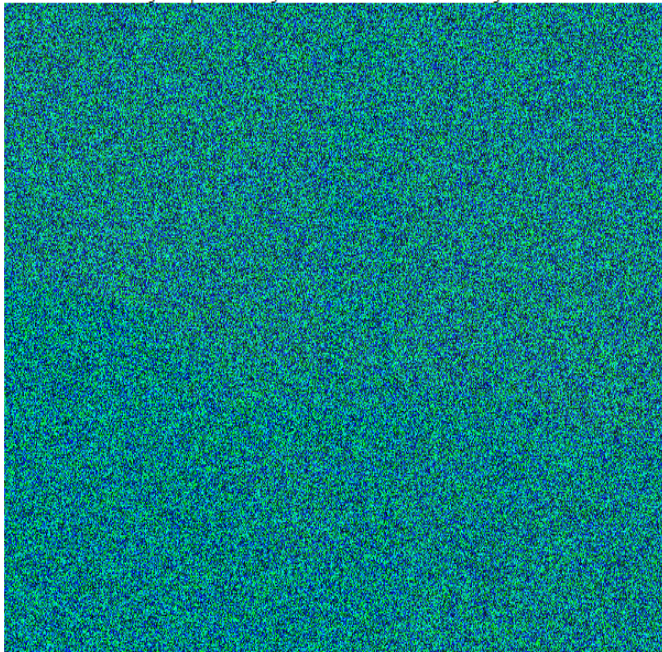
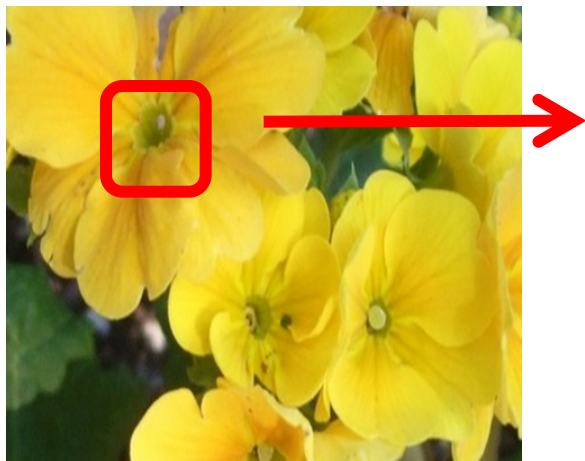


Image Representation in Computer

- Digital images everywhere
- Look natural, rounded
- Behind the scenes?
- lots of little numbers



Pixel 像素



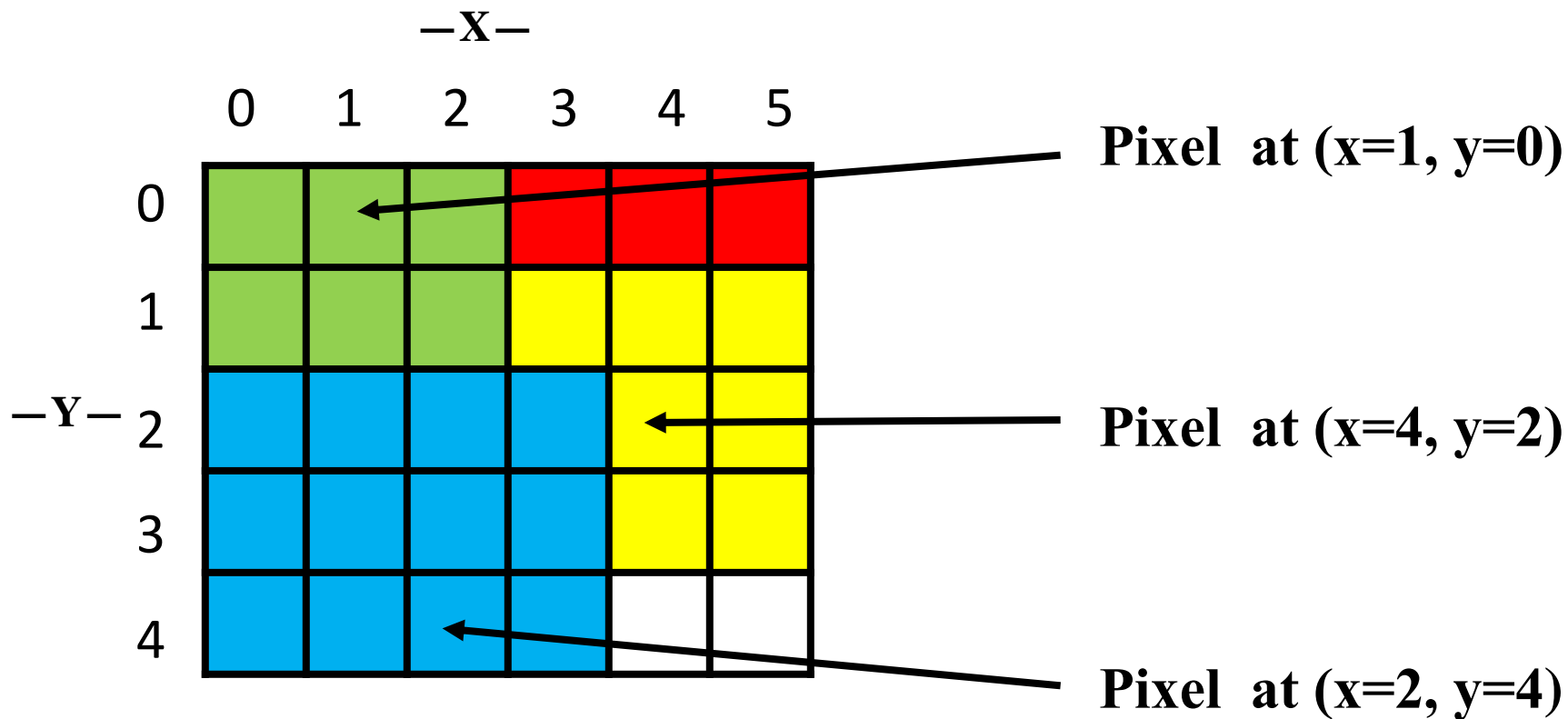
Definition

The word is a combination of pix, for picture, and element. In digital imaging, a pixel, pel, dots or picture element is a **physical point in a raster image, or the smallest addressable element in an all points addressable display device**; so it is the smallest controllable element of a picture represented on the screen.

- Each pixel shows a single color
- An 800 x 600 image has 480,000 pixels in all
- How the pixels are organized?



Image Coordinate 图像坐标 (x,y)



- We've known how the pixels are organized
- How can a pixel represent a color in computer?





RGB - RGB Color Scheme

Three Numbers

- Any color can be represented by combination of red/green/blue light.
- Any color can be represented by three numbers

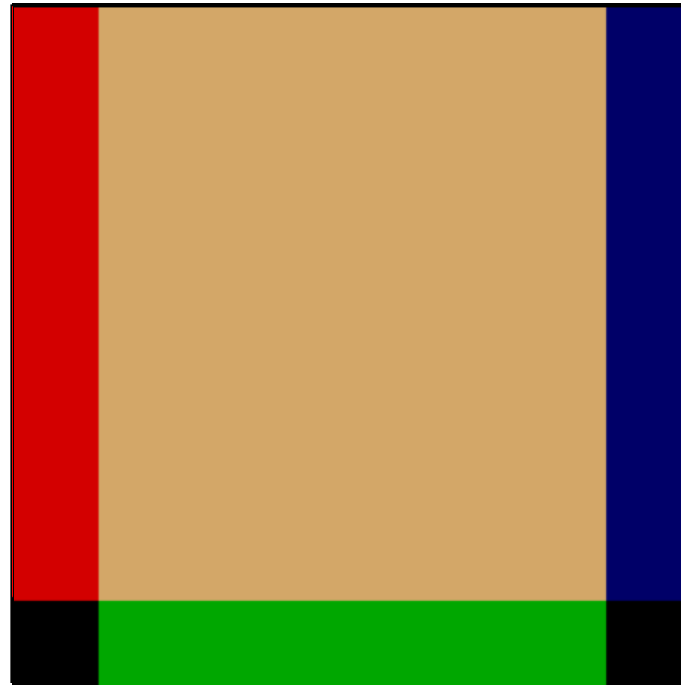
RGB Explorer

Red:

Green:

Blue:

Red: **255** Green: **165** Blue: **154**

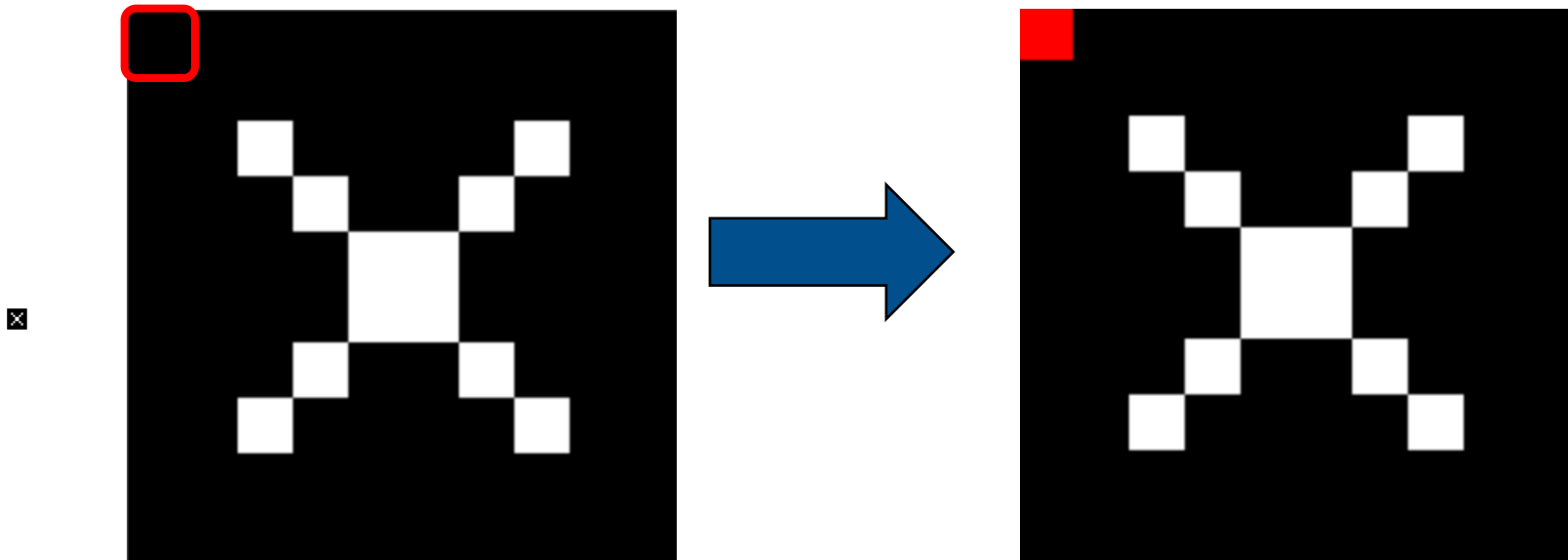




Digital Atomization (数字化)

- Complex whole - "atomized"
- Broken down into lots of little numbers (behind the scene)
- How to change images on computer?
- Change numbers that make up the image!!

An Image Example - Change a pixel's color



“x.png”
10*10 pixels

Definition

- In computer programming, a **function** is a sequence of program instructions that perform a specific task, packaged as a unit. In different programming languages, a function may be called a procedure, a subroutine, a method, or a subprogram.

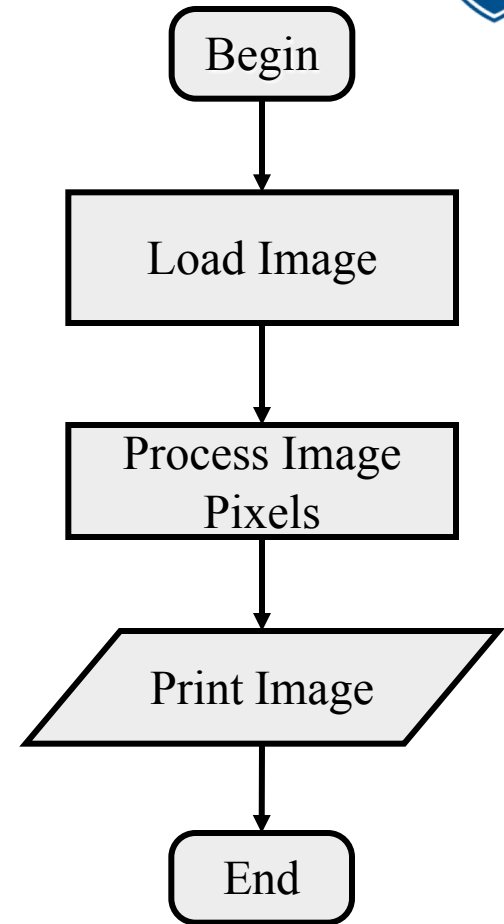
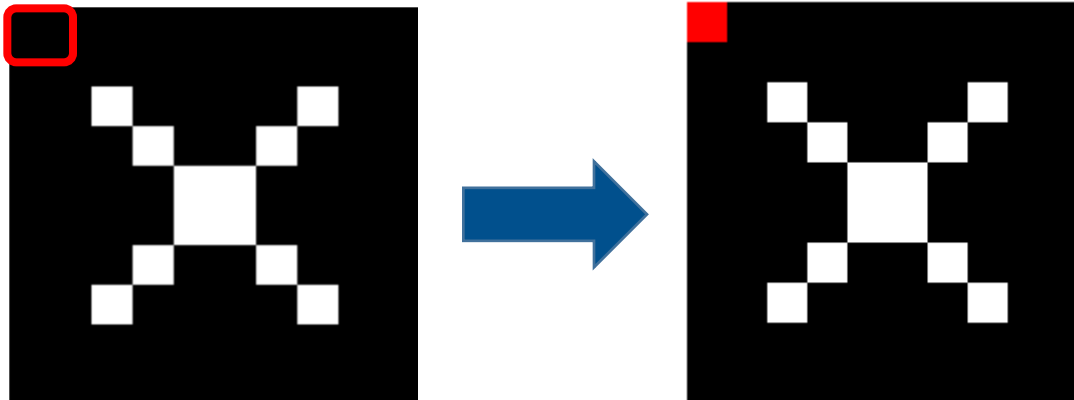
[Image Zoom Exercise](#)

getPixel() & setRed()

```
image = new SimpleImage("x.png");
```

```
pixel = image.getPixel(0, 0);  
pixel.setRed(255);
```

```
print(image);
```



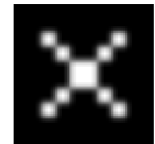
[pixel.setRed\(255\) Example](#)

Pixel Set Red/Green/Blue Functions

pixel.**setRed**(number); -- **set the red value** of the pixel to be the given number (0..255)

pixel.**setGreen**(number); -- **set the green value**

pixel.**setBlue**(number); -- **set the blue value**



Object.function() Pattern

Object:

In the [class-based object-oriented programming](#) (OOP) paradigm, "object" refers to a particular [instance](#) of a [class](#) where the object can be a combination of variables, functions, and data structures.

- One, we're seeing here is the **noun.verb()** pattern
- The noun as an [object](#) is the thing we want to operate on (e.g. a [pixel](#))
- The verb operates a [function](#) on that noun, like [setRed\(\)](#)

Image Function References

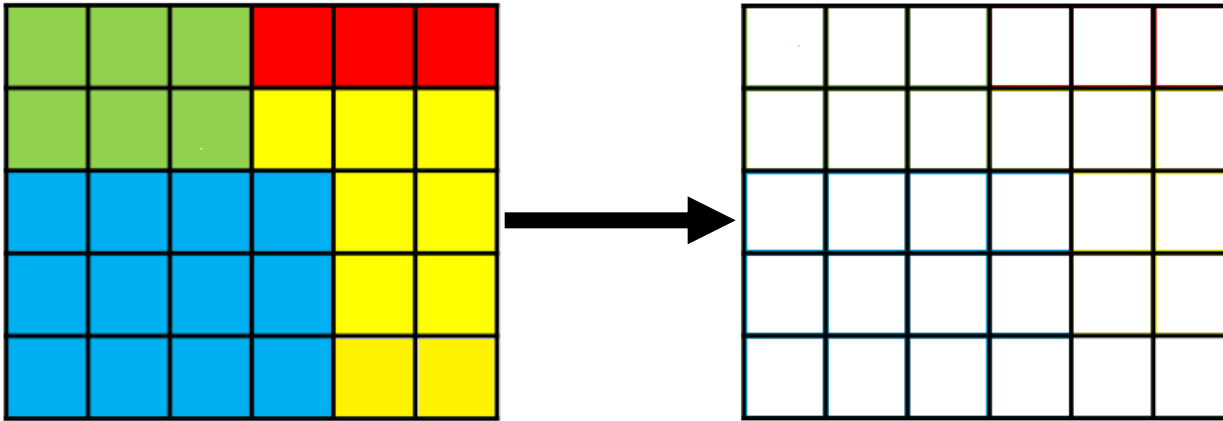
- A Link
 - <http://web.stanford.edu/class/cs101/image-functions-reference.html>
- [Experiments](#) on Pixel (0,0)



Definition

- A **loop** is a sequence of statements which is **specified once but which may be carried out several times in succession**. The code “inside” the loop is obeyed a specified number of times, or once for each of a collection of items, or until some condition is met, or indefinitely.

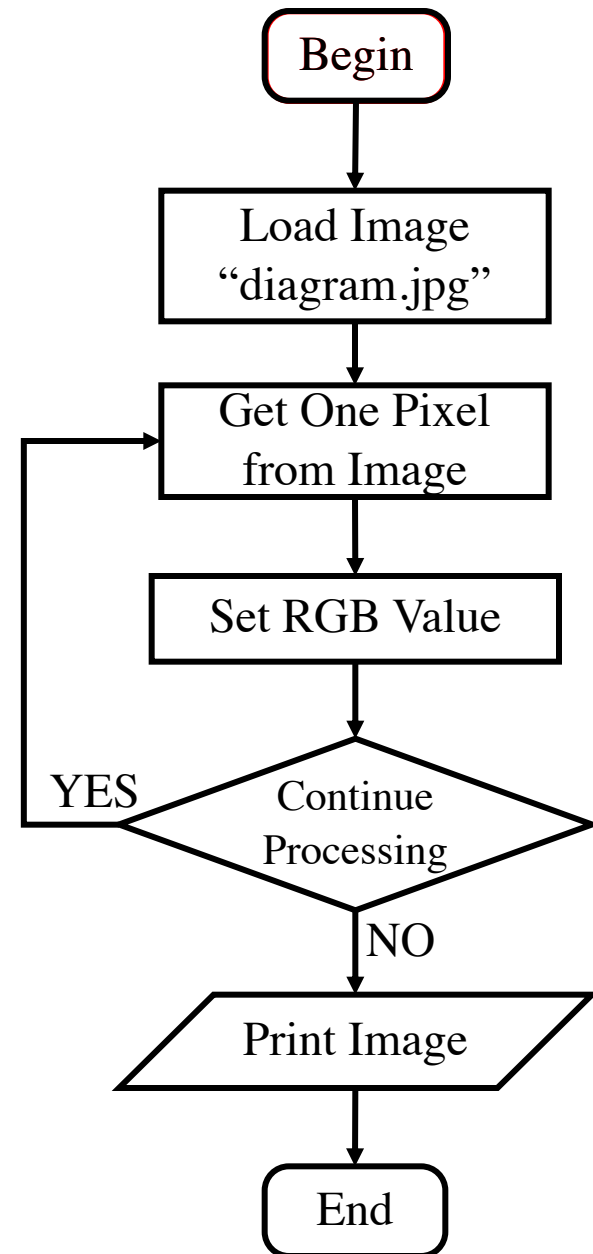
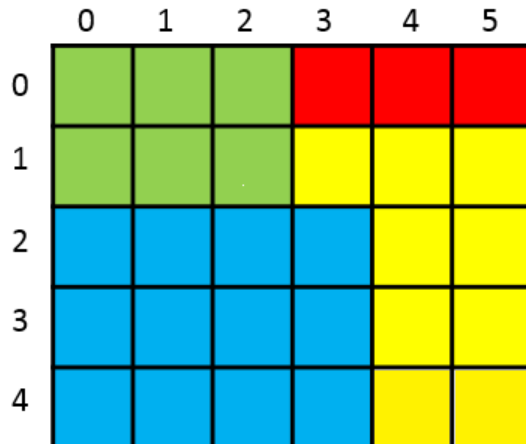
How can we change the color of all the pixels in image?



Loops (循环) do some steps "for all these items"



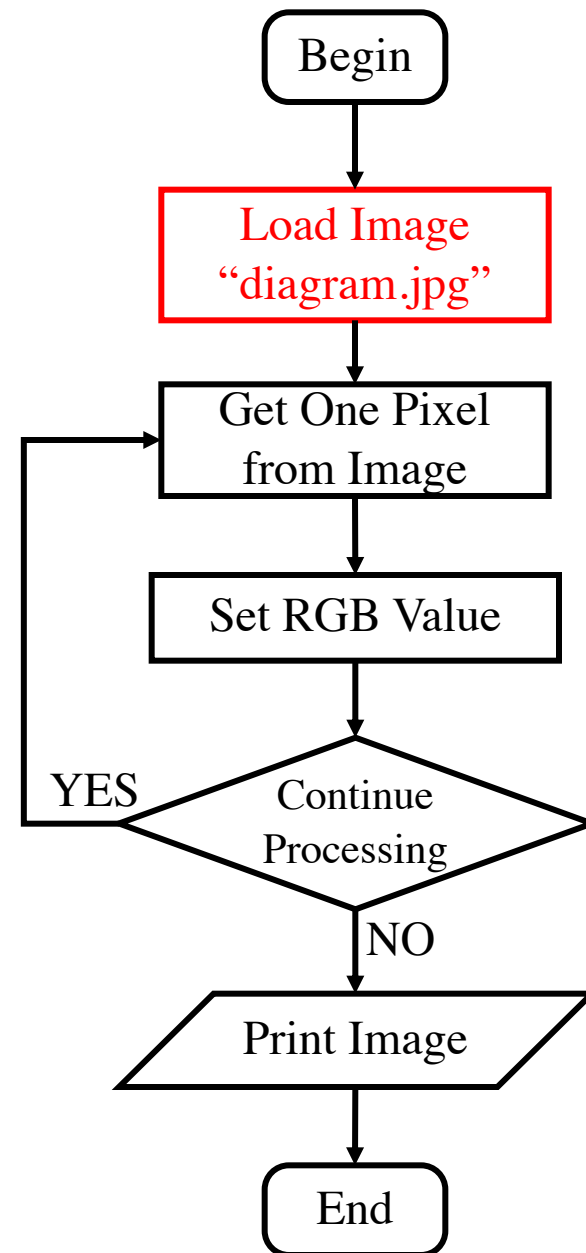
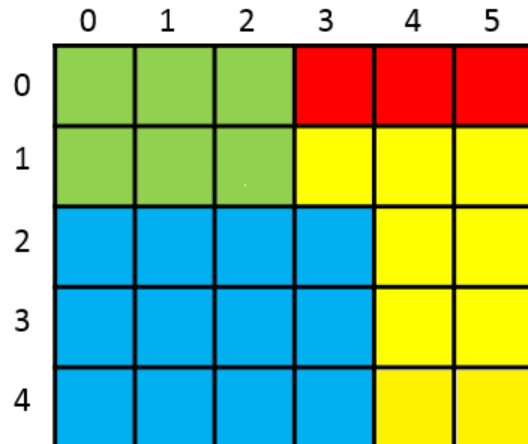
```
image = new SimpleImage("diagram.jpg");  
  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
  
print (image);
```



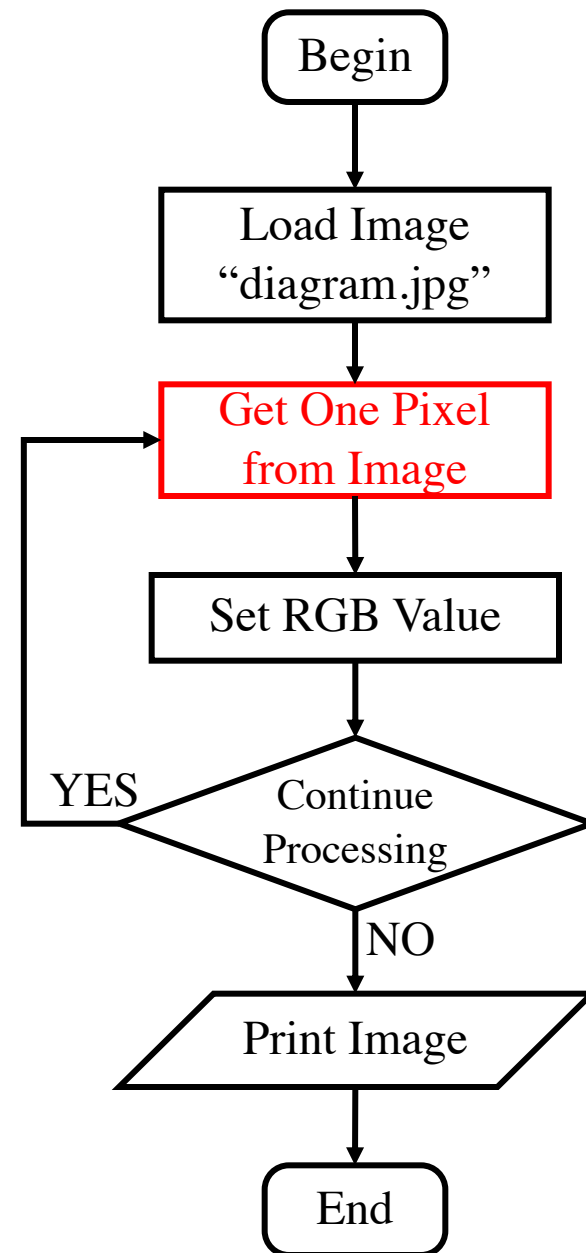
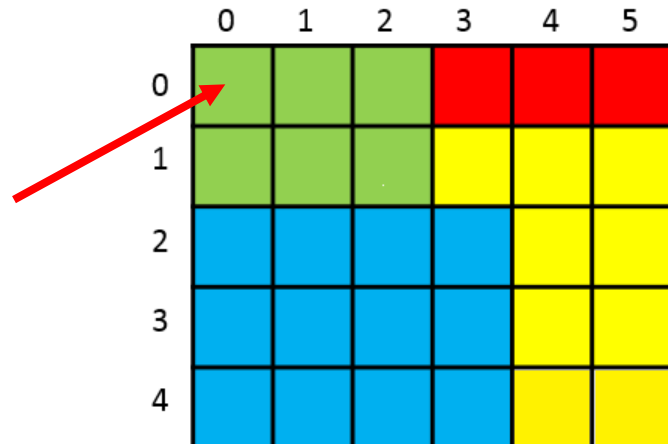
```
image = new SimpleImage("diagram.jpg");
```

```
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}
```

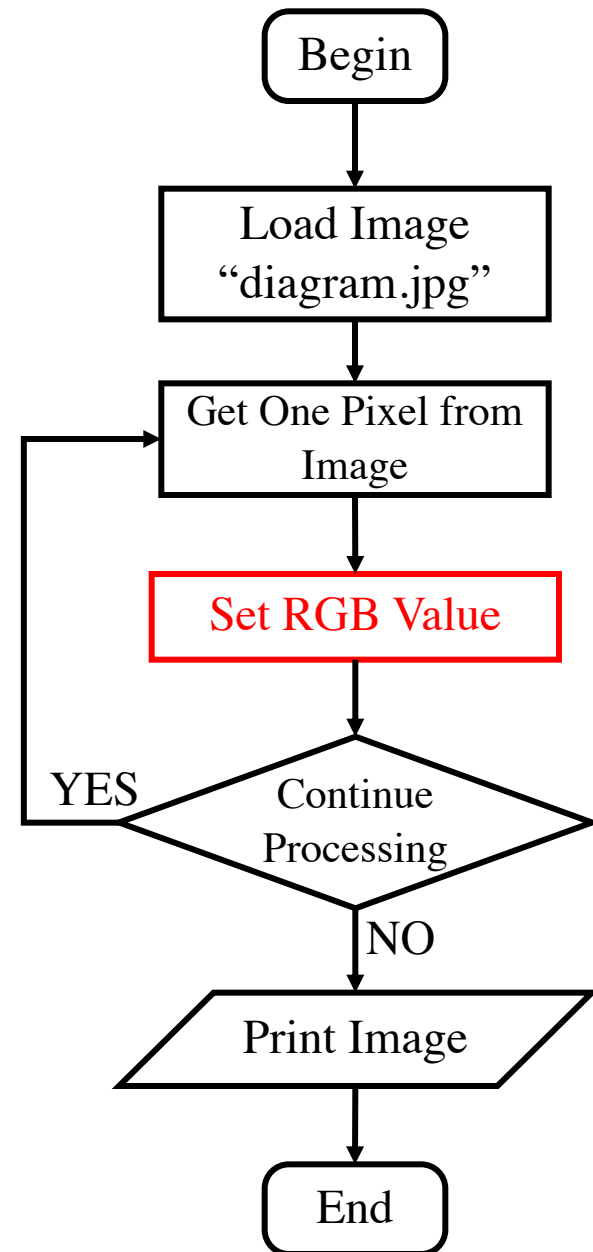
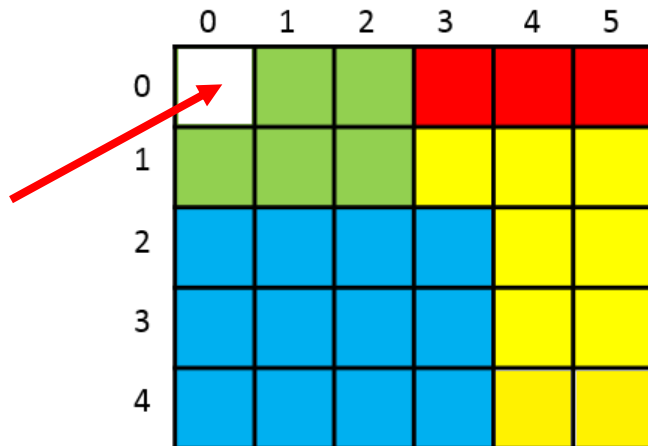
```
print (image);
```



```
image = new SimpleImage("diagram.jpg");  
  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
  
print (image);
```



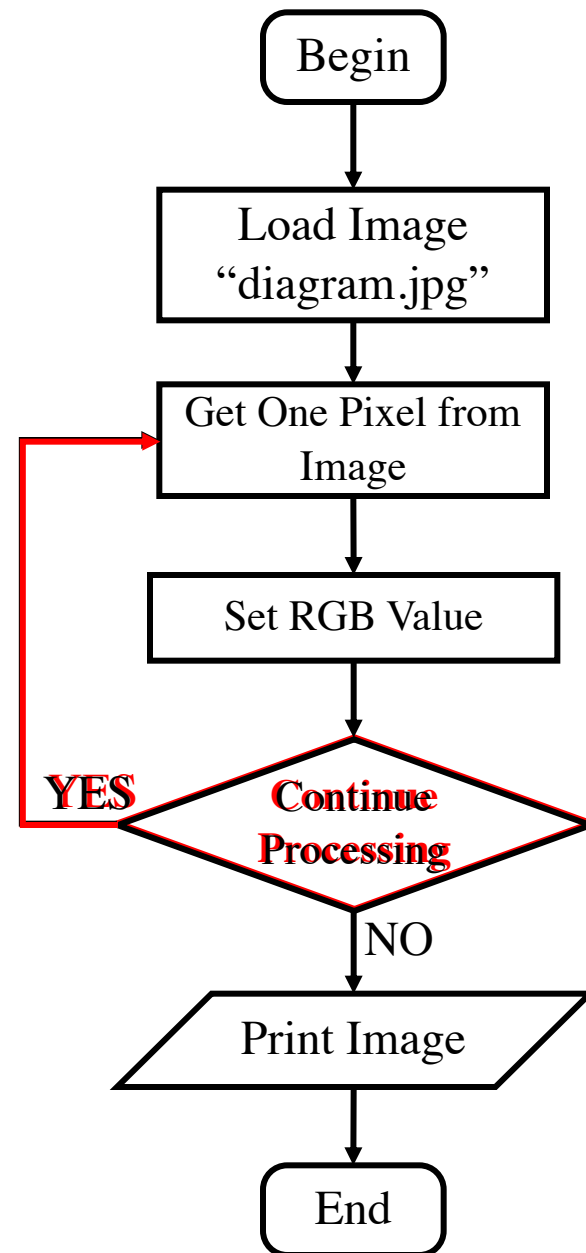
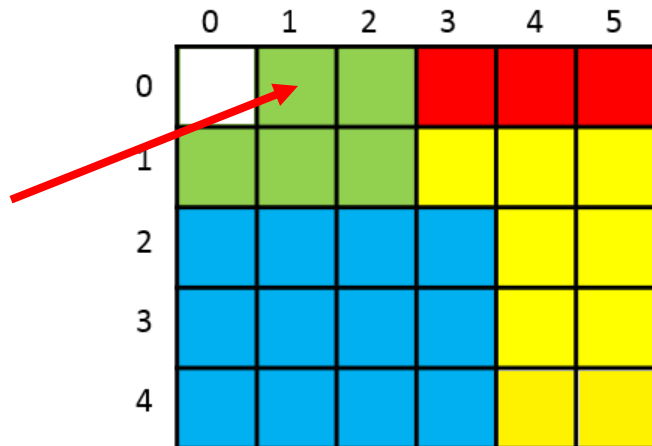
```
image = new SimpleImage("diagram.jpg");  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
print (image);
```



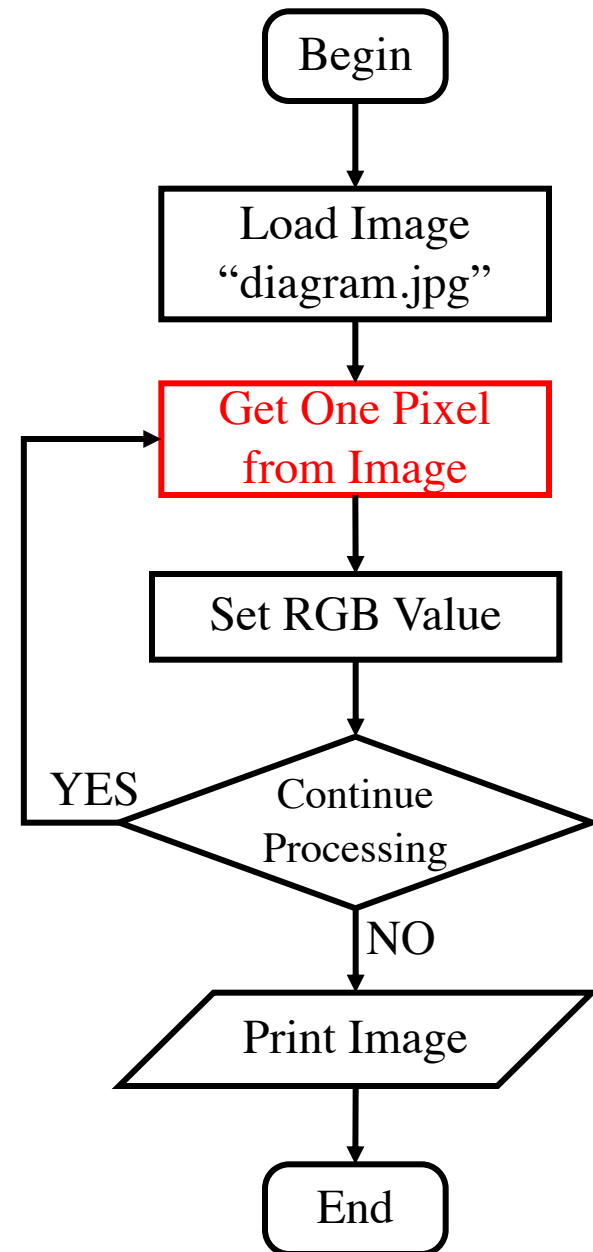
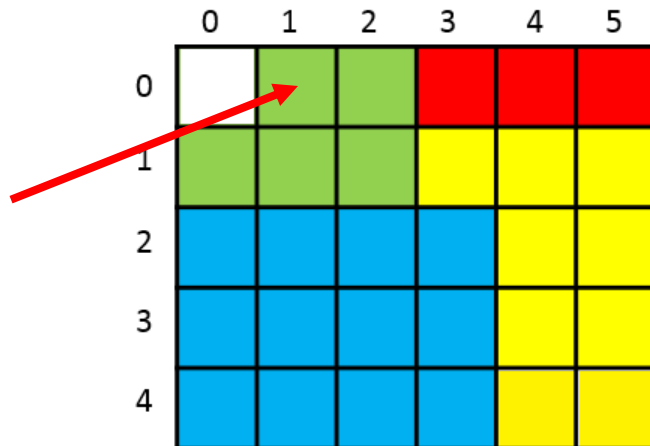
```
image = new SimpleImage("diagram.jpg");
```

```
for (pixel : image) {
    pixel.setRed(255);
    pixel.setGreen(255);
    pixel.setBlue(255);
}
```

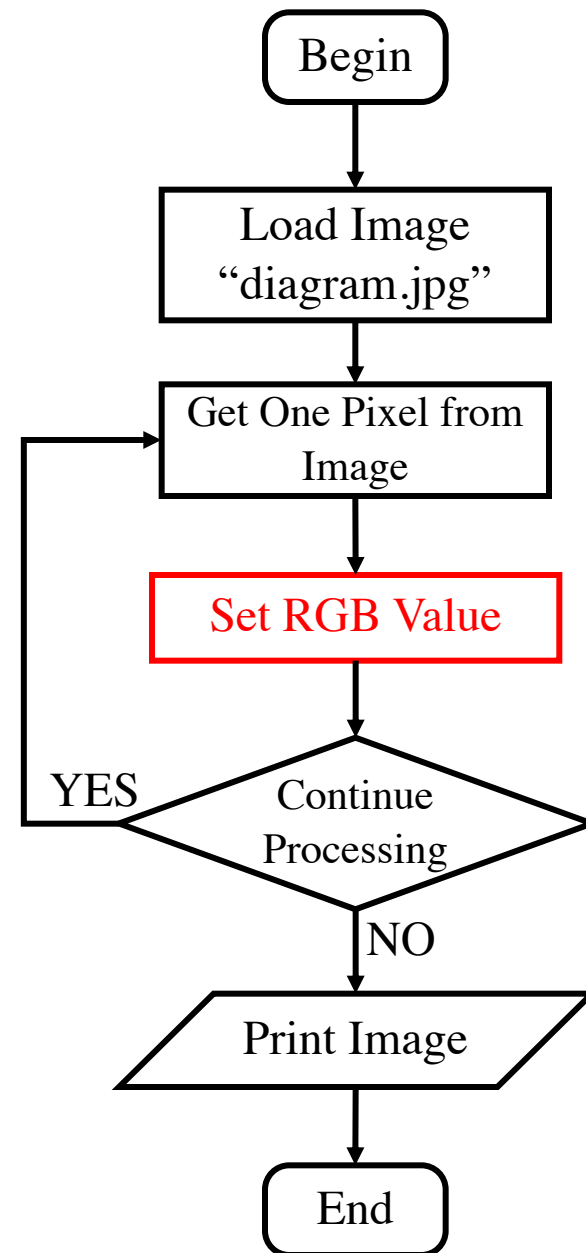
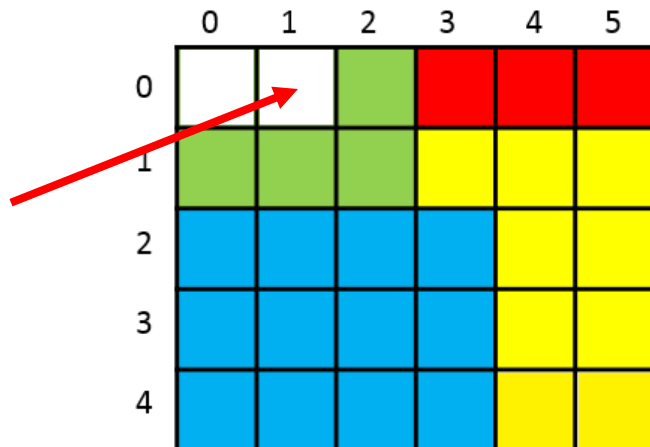
```
print (image);
```




```
image = new SimpleImage("diagram.jpg");  
  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
  
print (image);
```

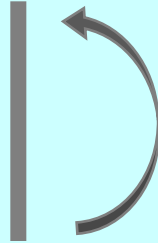


```
image = new SimpleImage("diagram.jpg");  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
print (image);
```



```
image = new SimpleImage("diagram.jpg");
```

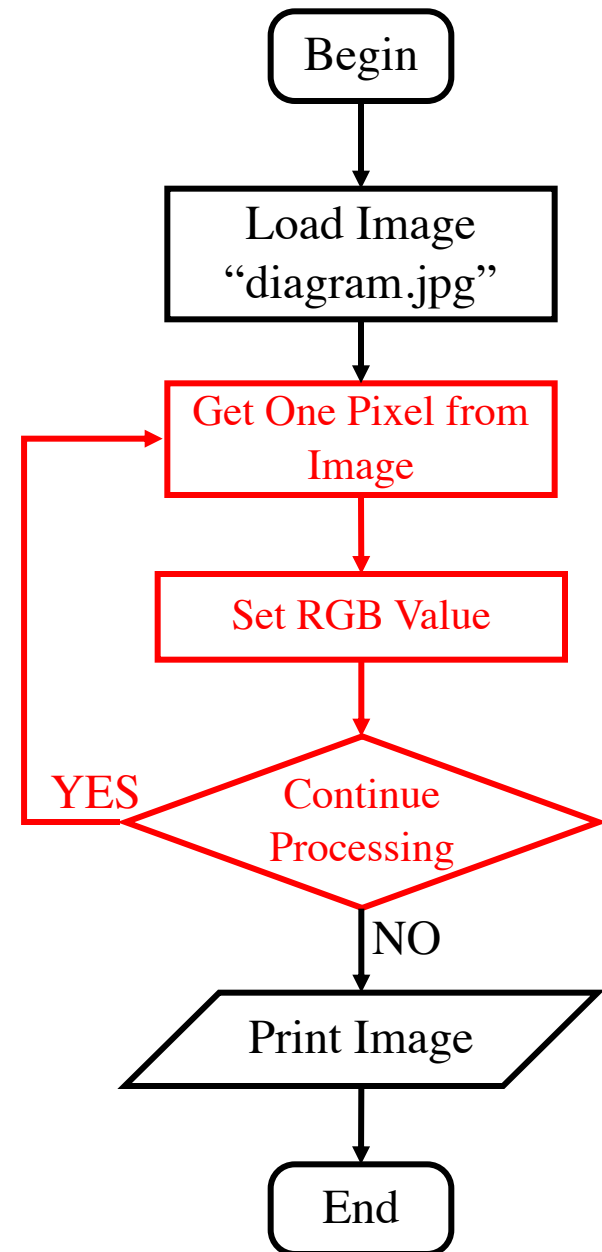
```
for (pixel : image) {
    pixel.setRed(255);
    pixel.setGreen(255);
    pixel.setBlue(255);
}
```



```
print (image);
```

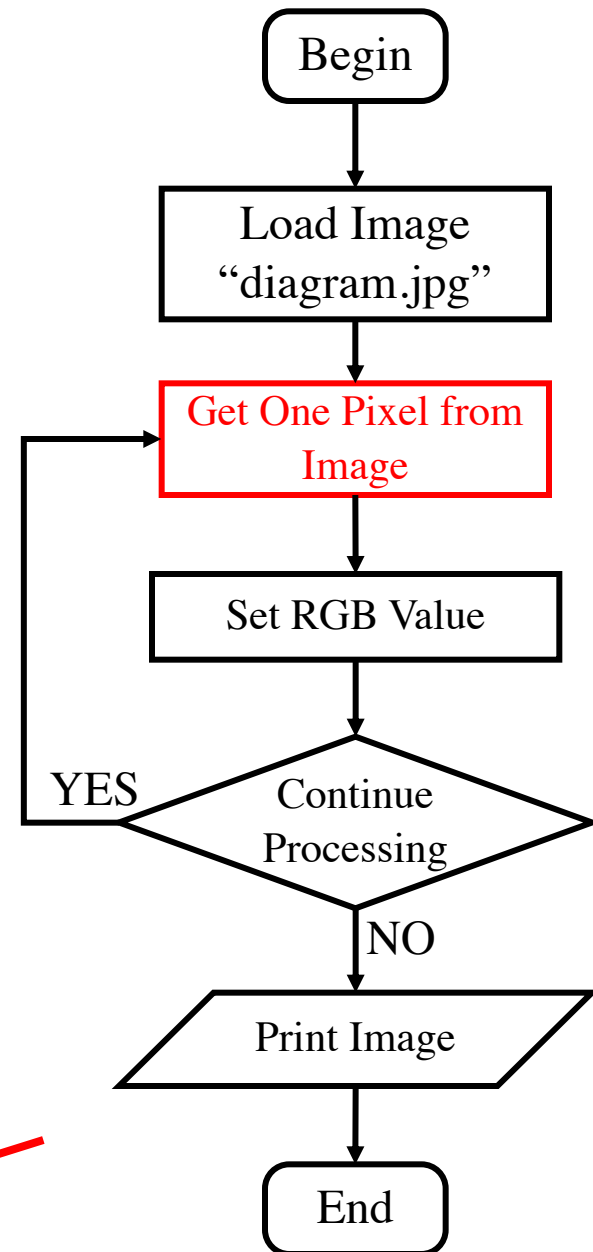
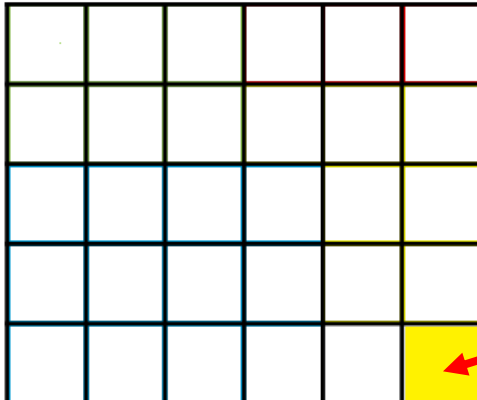
**computer runs it again and again,
once for each pixel**

	0	1	2	3	4	5
0						
1						
2						
3						
4						



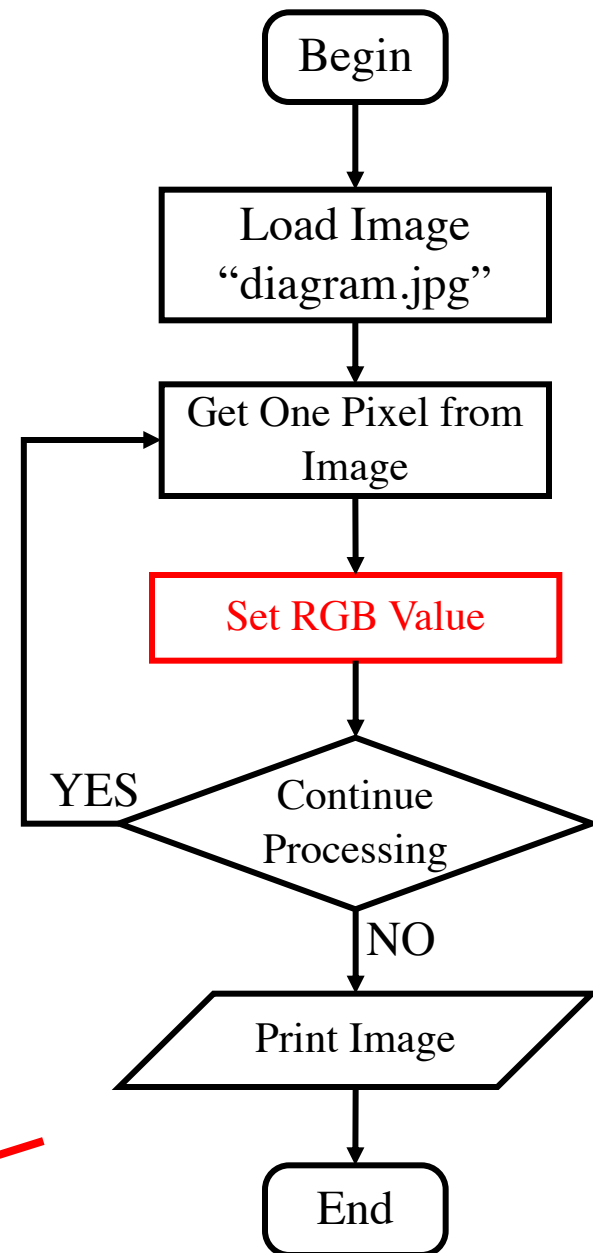

```
image = new SimpleImage("diagram.jpg");  
  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
  
print (image);
```

	0	1	2	3	4	5
0						
1						
2						
3						
4						



```
image = new SimpleImage("diagram.jpg");  
for (pixel : image) {  
    pixel.setRed(255);  
    pixel.setGreen(255);  
    pixel.setBlue(255);  
}  
print (image);
```

	0	1	2	3	4	5
0						
1						
2						
3						
4						



```

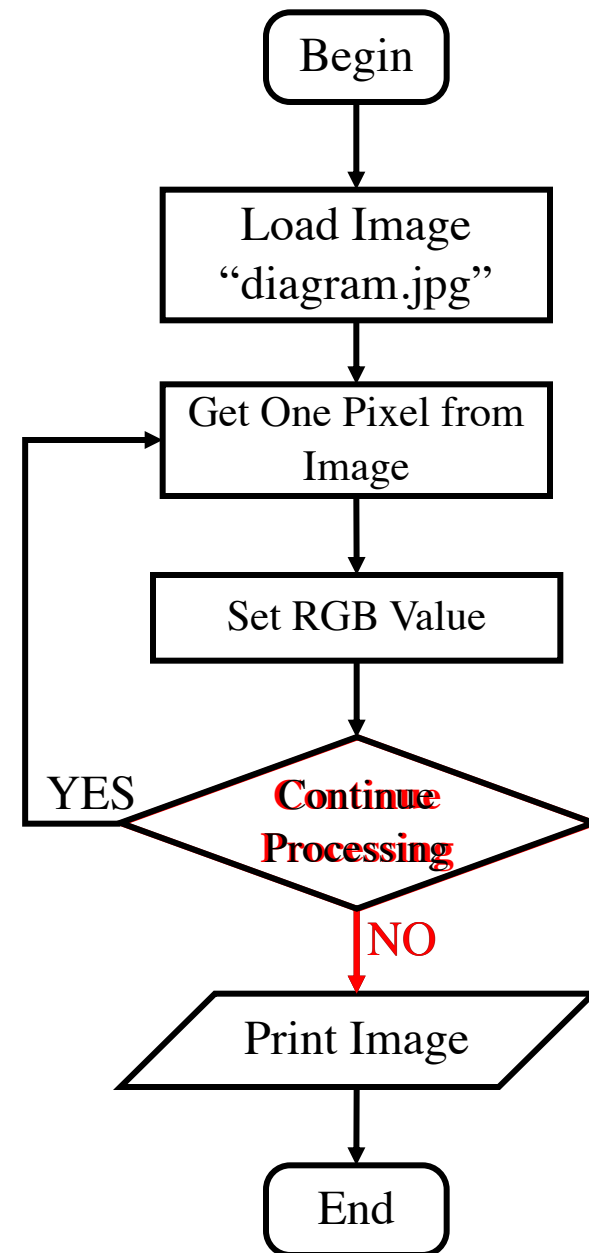
image = new SimpleImage("diagram.jpg");

for (pixel : image) {
    pixel.setRed(255);
    pixel.setGreen(255);
    pixel.setBlue(255);
}

print (image);

```

	0	1	2	3	4	5
0						
1						
2						
3						
4						



```

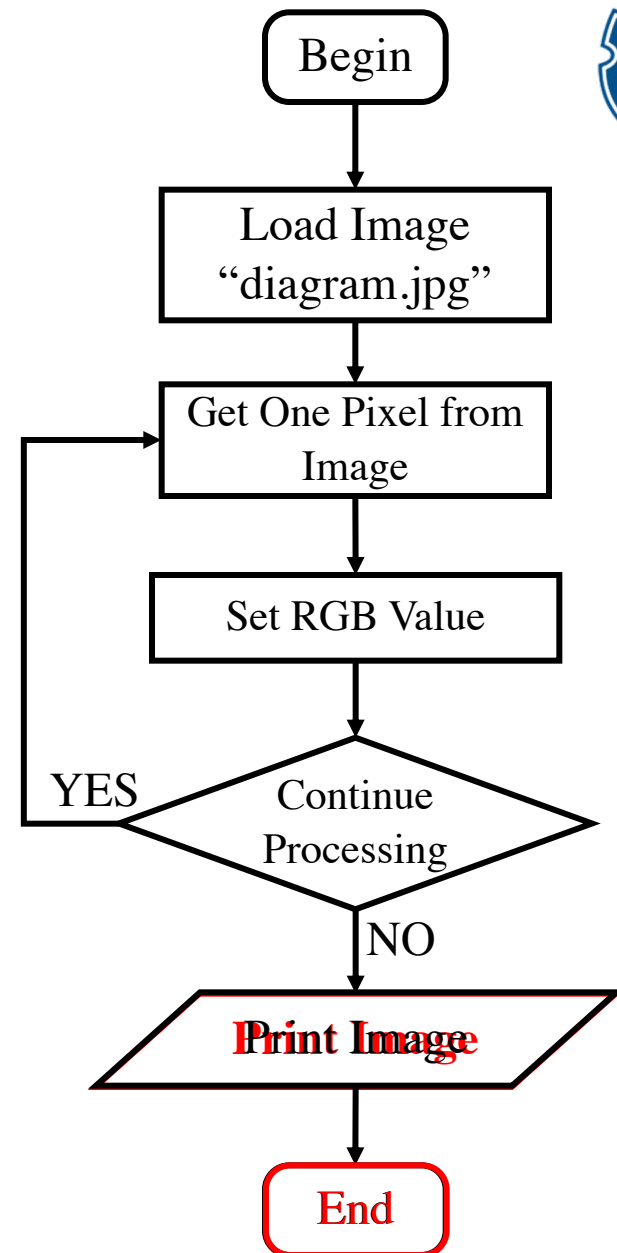
image = new SimpleImage("diagram.jpg");

for (pixel : image) {
    pixel.setRed(255);
    pixel.setGreen(255);
    pixel.setBlue(255);
}

print (image);

```

	0	1	2	3	4	5
0						
1						
2						
3						
4						



For-Loop Example

```
image = new SimpleImage("puzzle.jpg");
```

```
for (pixel: image) {
```

```
    pixel.setRed(255);
```

```
    pixel.setGreen(255);
```

```
    pixel.setBlue(0);
```

```
}
```

```
print(image);
```



change to
yellow





How Does That Loop Work?

- For-loop syntax a little weird, but it's the same every time
- When we want "for every item do this", we'll use a for-loop
- For-loop has 2 parts
 - `for (pixel: image) {` - start the for-loop
 - `"body" lines of code within curly braces { .. }`
- A powerful feature, we specify a few lines of code, computer takes care of running them thousands of times
- Computer = powerful + stupid theme
- Code inside the loop is special: run it for every pixel

[Loop Exercise](#)



e.g:

$3+5-9$

$8*2/4$

$(2+6)/4$

Expression (表达式)

An **expression** in a programming language is a combination of one or more explicit values, constants, variables, operators, and functions that the programming language interprets (according to its particular rules of precedence and of association) and computes to produce (“to return”, in a stateful environment) another value.



Expression: $1 + 1$

- an "expression" written in the code like $11 + 13$ computes the value to use. For example you could write something like this:

```
print(11 + 31);
```



Set/Get Pattern:

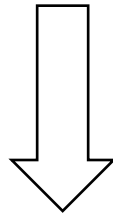
combine `pixel.setRed()` and `pixel.getRed()`

- `pixel.getRed()` -- retrieves the red value from a pixel
- `pixel.getGreen()` -- retrieves the green value
- `pixel.getBlue()` -- retrieves the blue value



- Suppose we want to **double** the red value of a pixel
- A more realistic operation -- relative

```
// Doubles the pixel's red value  
old = pixel.getRed();  
pixel.setRed(old * 2);
```



```
pixel.setRed(pixel.getRed() * 2);
```

Image Expression Exercise



Loops With Expressions - Set/Get Patterns

Now can express **relative** color number changes

e.g. double the red value (with set/get pattern):

```
pixel.setRed(pixel.getRed() * 2);
```

e.g. halve the red value:

```
pixel.setRed(pixel.getRed() * 0.5);
```

Bottom line, we'll use this pattern a lot:

```
pixel.setRed(pixel.getRed() * something);
```

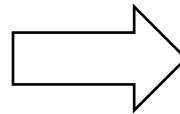


5-10-20 Image Puzzles

- 5-10-20 Puzzles
- Image where red, green, and blue divided by 5, 10, or 20
- Scale them back up by $\times 5$, $\times 10$, $\times 20$ to recover the original image
- Puzzle: don't know which number goes with which color
- Experiment to figure it out
- There are only 6 possibilities:
- 5 10 20, 5 20 10, 10 5 20, 10 20 5, 20 5 10, 20 10 5
- i.e. 5 first $\times 2$, 10 first $\times 2$, 20 first $\times 2$

5-10-20 Banana Puzzle

- Yellow banana, background of dark red bricks
- Bits of dark green moss between the bricks
- Use code pattern: `pixel.setRed(pixel.getRed() * 5);`
- Scale up all three colors, using factors 5, 10, 20 to fix the image



Let's solve puzzles !

What's the real value of red color?
How to generate puzzle image?

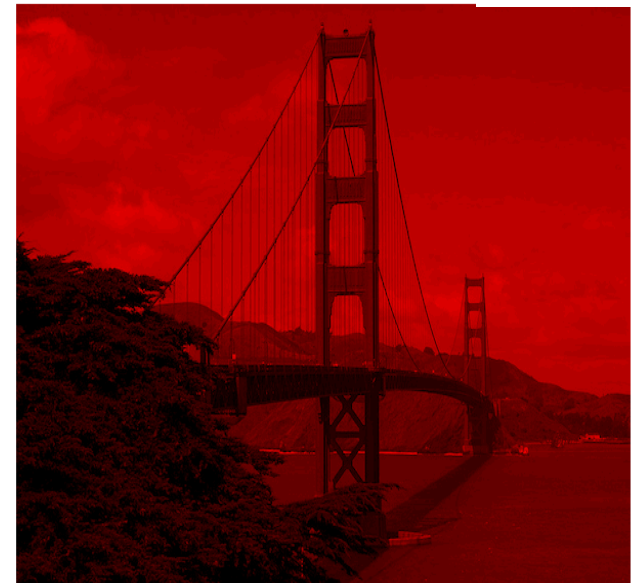
- For each pixel, set green and blue to 0
- For each pixel, expand red to 10x



```
image = new SimpleImage("puzzle.jpg");
```

```
for (pixel: image) {  
    pixel.setGreen(0);  
    pixel.setBlue(0);  
    pixel.setRed(pixel.getRed() * 10);  
}
```

```
print(image);
```

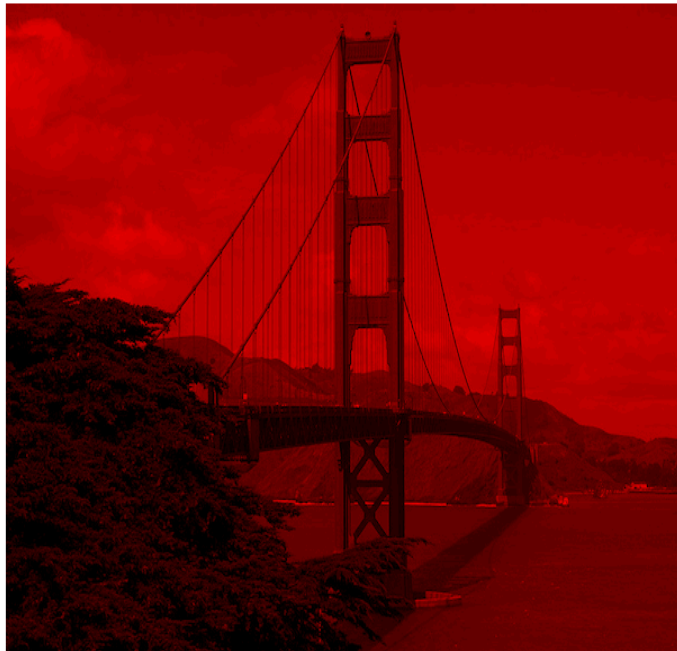


Puzzles Exercise

Grayscale (灰度图)



Here we have the grayscale image



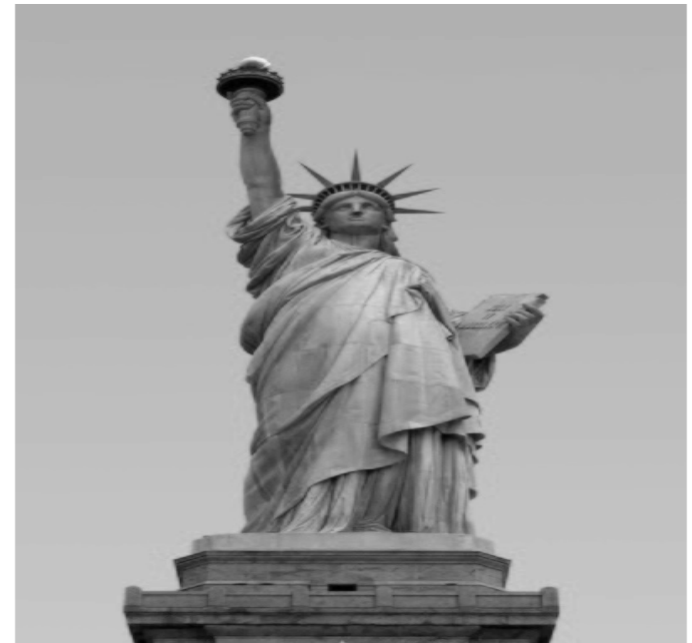
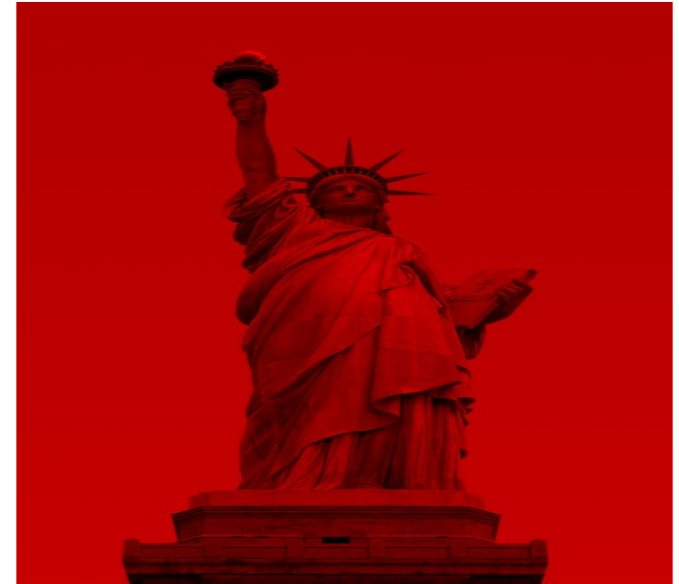
we saw this sort of image in an earlier puzzle solution

This is an image of the Statue of Liberty where all of the data is in the red values

The green and blue values are all zero.

This image looks quite wrong.

We want to make the image grayscale, not just red.





Grayscale

- In photography and computing, **a grayscale or greyscale digital image** is an image in which **the value of each pixel is a single sample**, that is, it carries only intensity information. Images of this sort, also known as **black-and-white**, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.



- Visit the [RGB explorer](#)
- Figure out how to make a shade of gray
- e.g. RGB values to make: dark gray, medium gray, light gray

Examples of gray colors in RGB:

red	green	blue	color
50	50	50	dark gray
120	120	120	medium gray
200	200	200	light gray
In fact, the original pure black/white colors fit this all-equal pattern too, just using the values 0 and 255.			
0	0	0	pure black
255	255	255	pure white



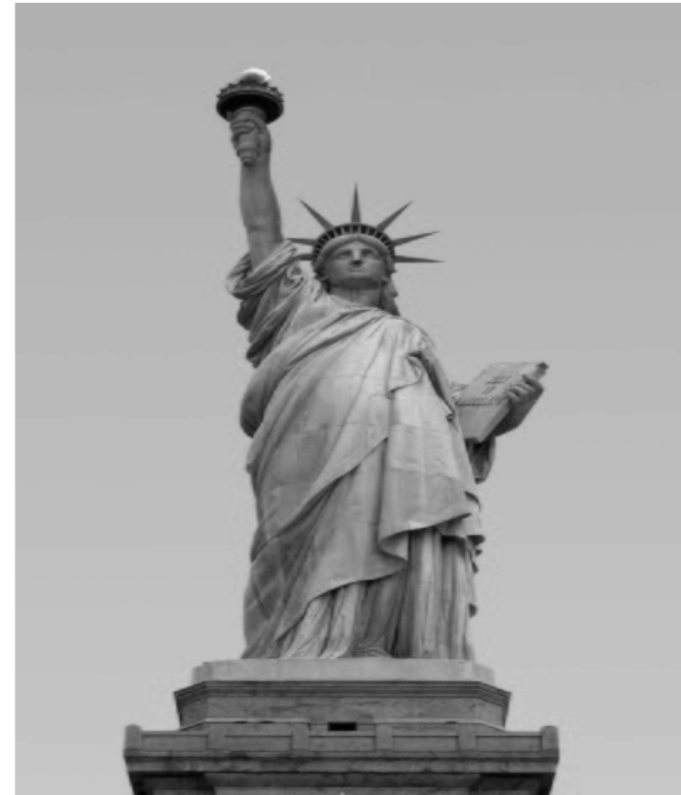
Algorithm

- Algorithm is a set of clear ordered steps, which produces results and terminates in finite time.

Algorithm

- For each pixel, set green and blue to the value of red

```
image = new SimpleImage("liberty-red.jpg");  
for (pixel: image) {  
    // your code here  
    pixel.setGreen(pixel.getRed());  
    pixel.setBlue(pixel.getRed());  
}  
print(image);
```





a regular color
image

How to convert a regular color
image to grayscale?

Red?



Green?



Blue?





For each pixel, how dark/light is it?

Look at just red or blue or green in isolation

	red	green	blue
pixel-1	200	50	50
pixel-2	0	75	75
pixel-3	100	250	250

How to decide which pixel below
is brightest? darkest?

The **average** combines and summarizes
the three values into one number 0..255.

	red	green	blue	average
				$\text{average} = (\text{red} + \text{green} + \text{blue}) / 3$
pixel-1	200	50	50	100 (medium bright)
pixel-2	0	75	75	50 (darkest)
pixel-3	100	250	250	200 (brightest)

The average is simple and works
fine for our purposes.

Algorithm

Grayscale Conversion Exercise

- For each pixel, add red+green+blue, then divide by 3
- For each pixel, store the average in a variable "avg"
- For each pixel, set the three colors' value to avg

```
image = new SimpleImage("flowers.jpg");  
for (pixel: image) {  
    // your code here  
  
    avg = (pixel.getRed() + pixel.getGreen() +  
          pixel.getBlue())/3;  
    pixel.setRed(avg);  
    pixel.setGreen(avg);  
    pixel.setBlue(avg);  
}  
print(image);
```



Summary



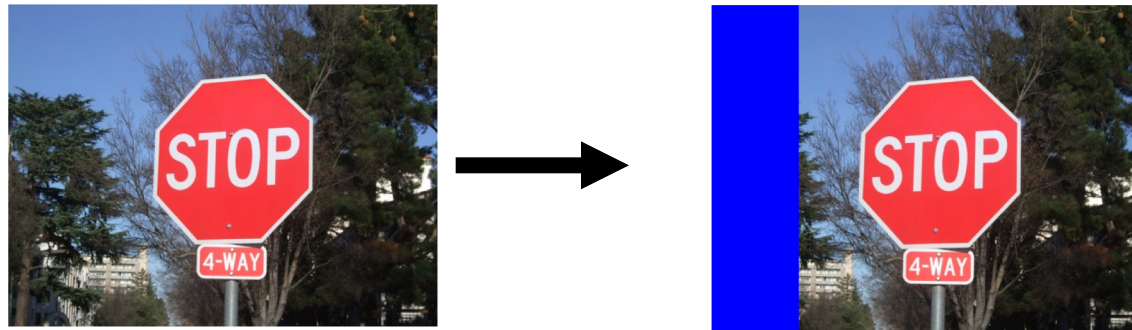
- Image Representation in Computer
 - Pixels in Image
 - Coordinates to locate pixels
 - RGB to represent image

- Basic Codes for Image Processing
 - Function to take an action on an object
 - Loop to do operation for the whole image
 - Expression – continuing in the next class

- Problem Solving



How can we set a vertical stripe 100 pixels wide at the far left to blue?

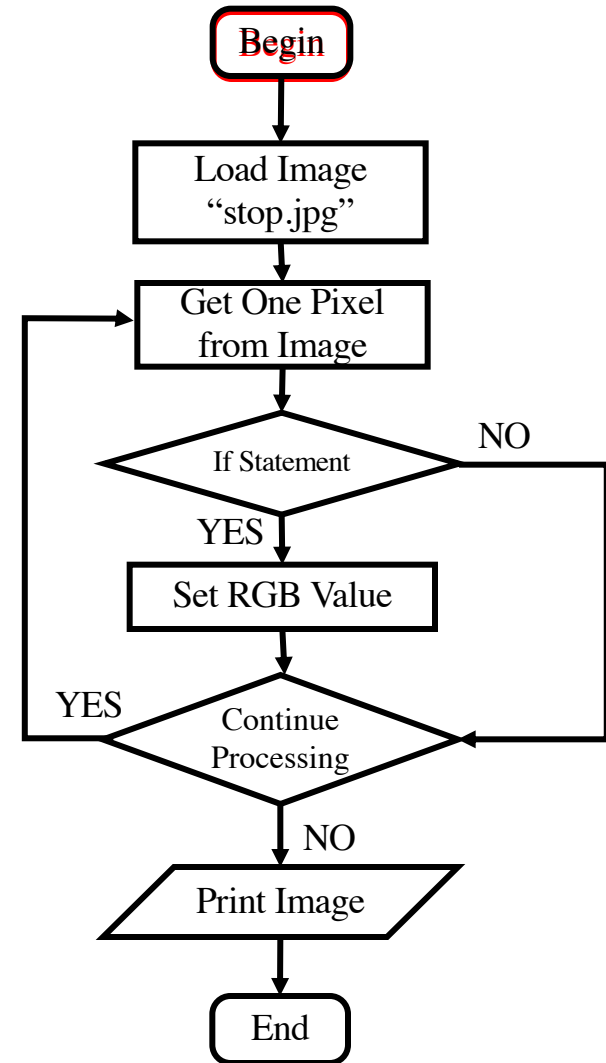


Bulk operation : Loops do some steps "for all these items"

Logic test true/false : Run some code only if a test is true



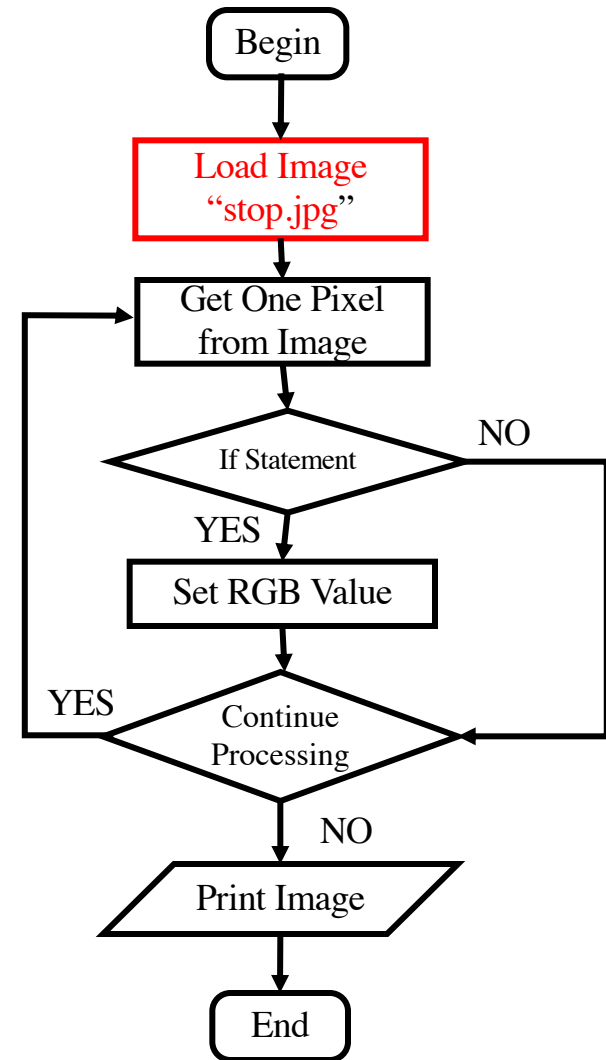
```
image = new SimpleImage("stop.jpg");  
  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
  
print (image);
```



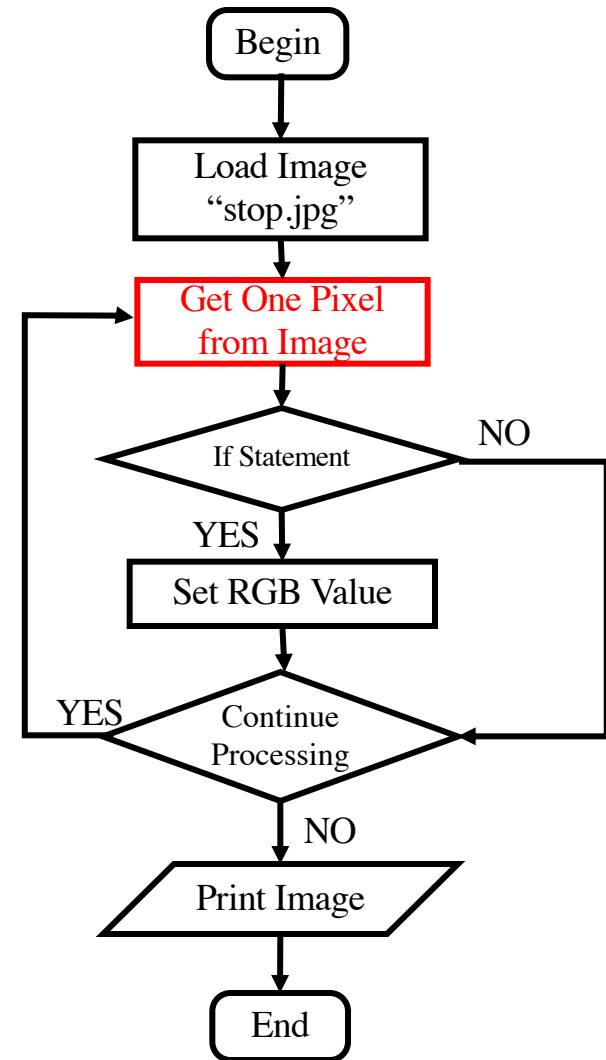
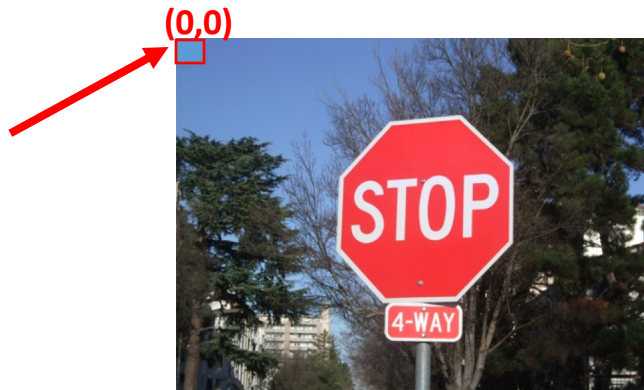
```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}
```

```
print (image);
```



```
image = new SimpleImage("stop.jpg");  
  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
  
print (image);
```



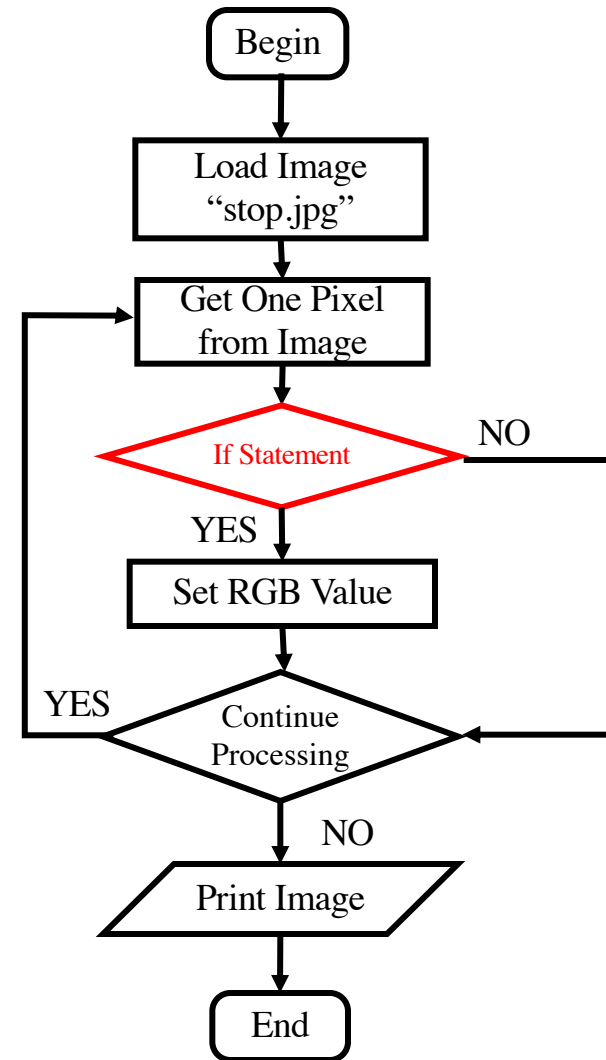
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



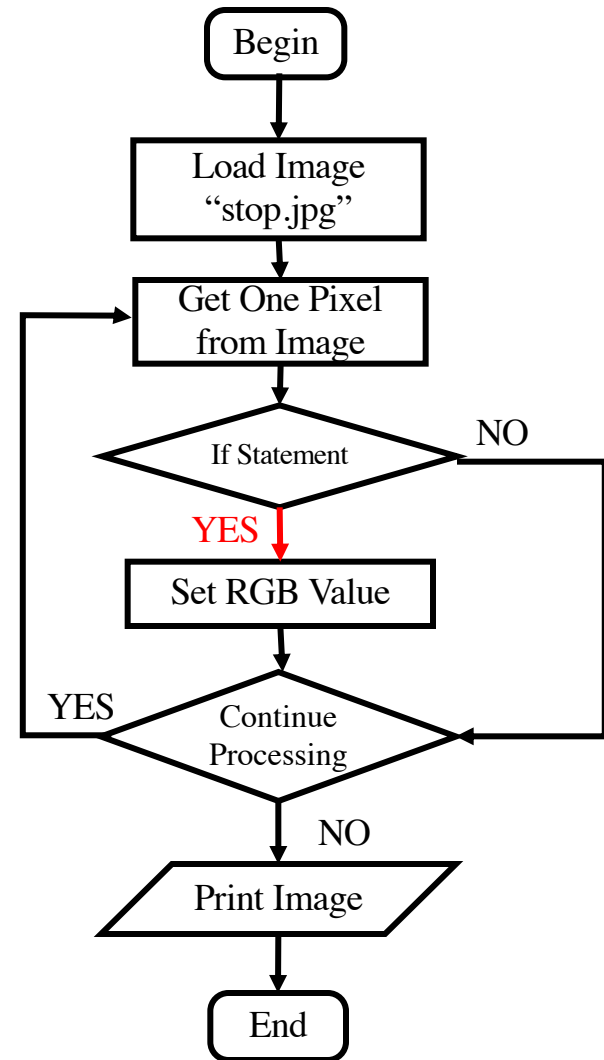
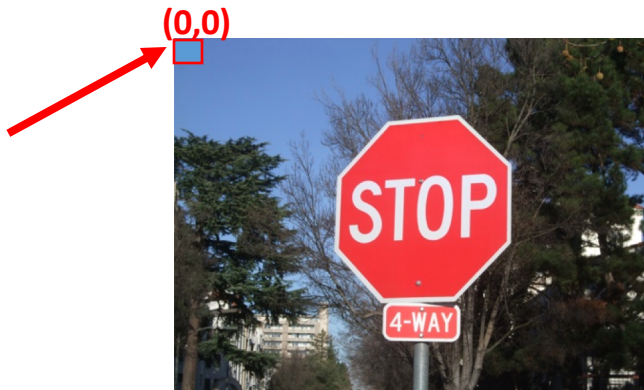

```

image = new SimpleImage("stop.jpg");

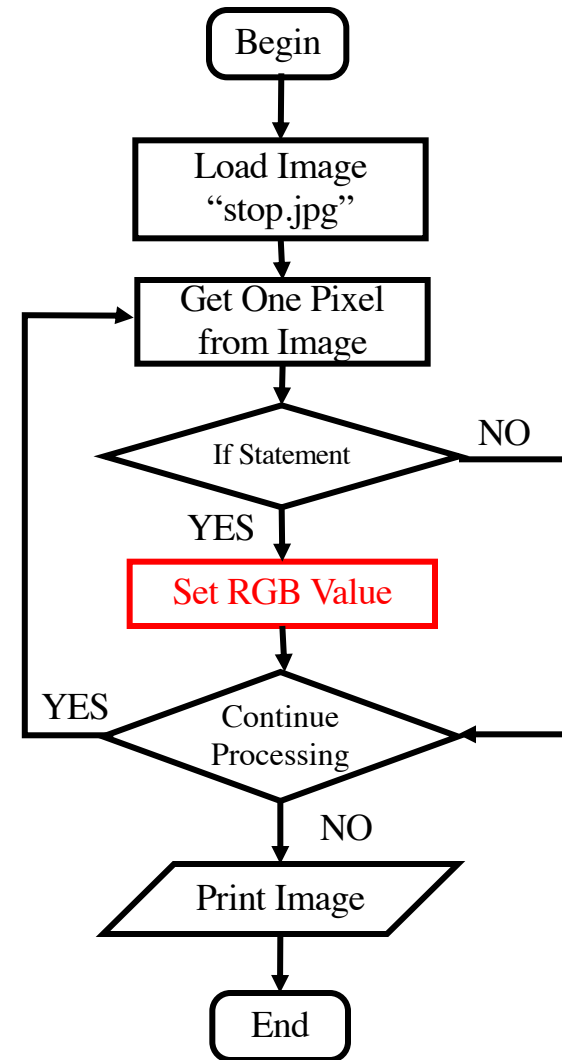
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



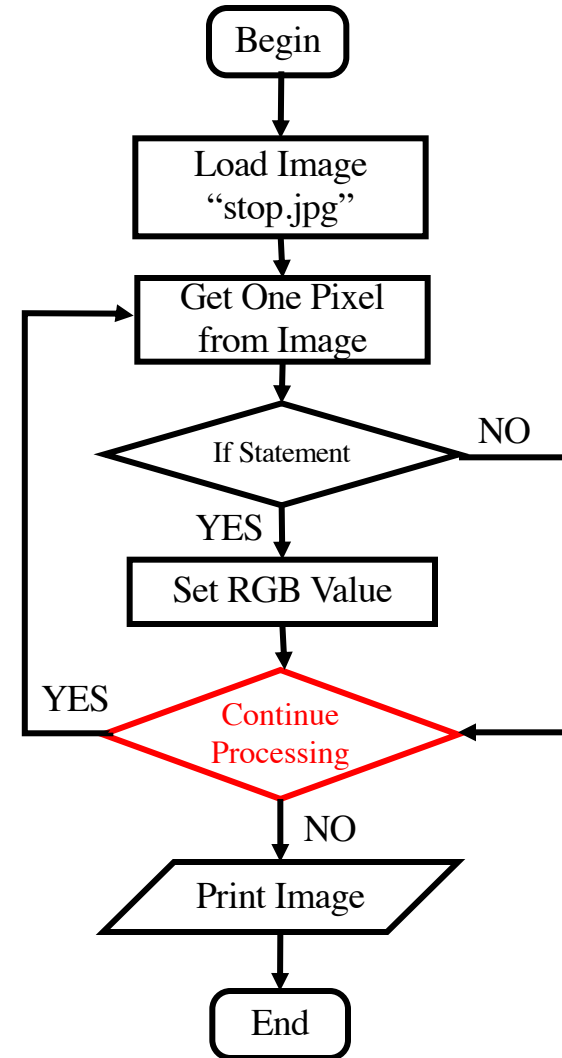
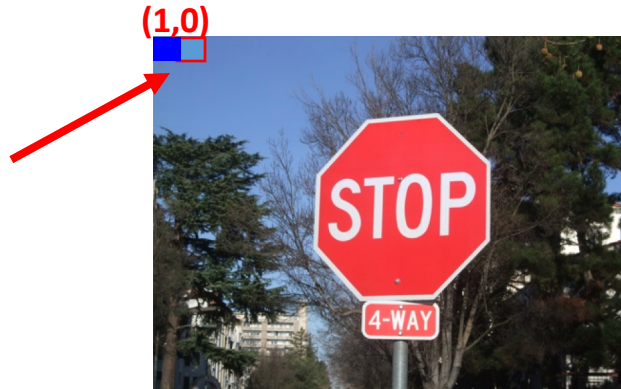
```
image = new SimpleImage("stop.jpg");  
  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
  
print (image);
```



```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}
```

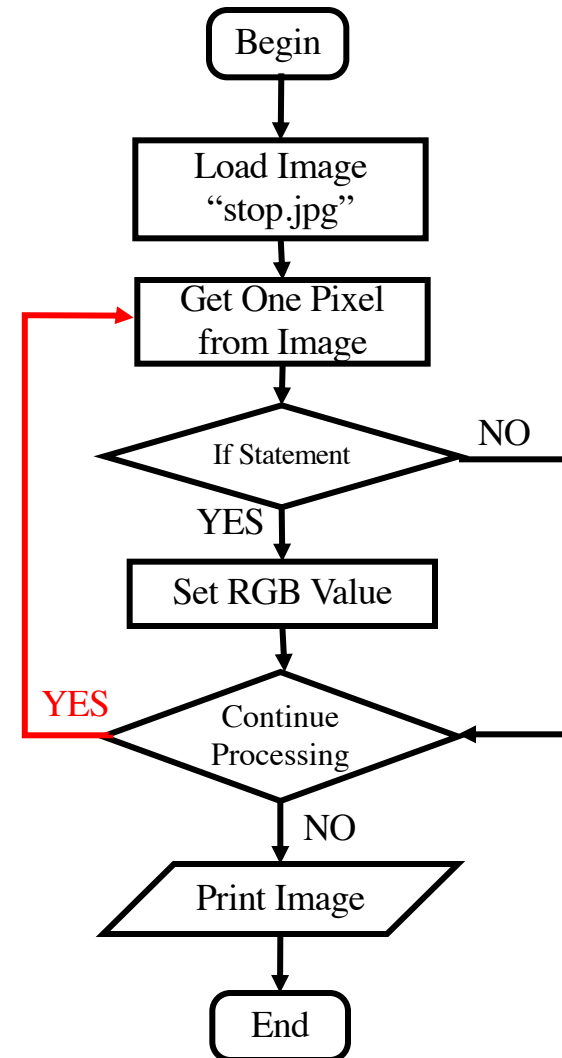
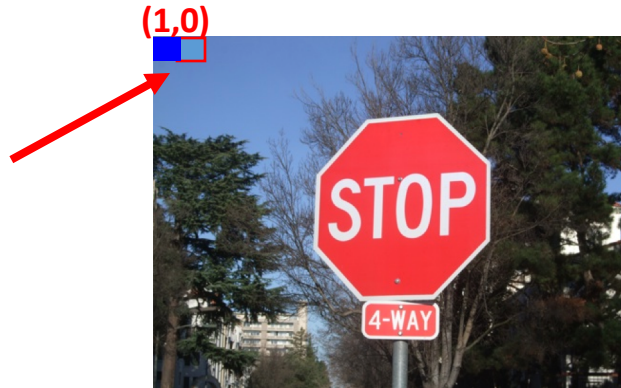
```
print (image);
```



```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}
```

```
print (image);
```



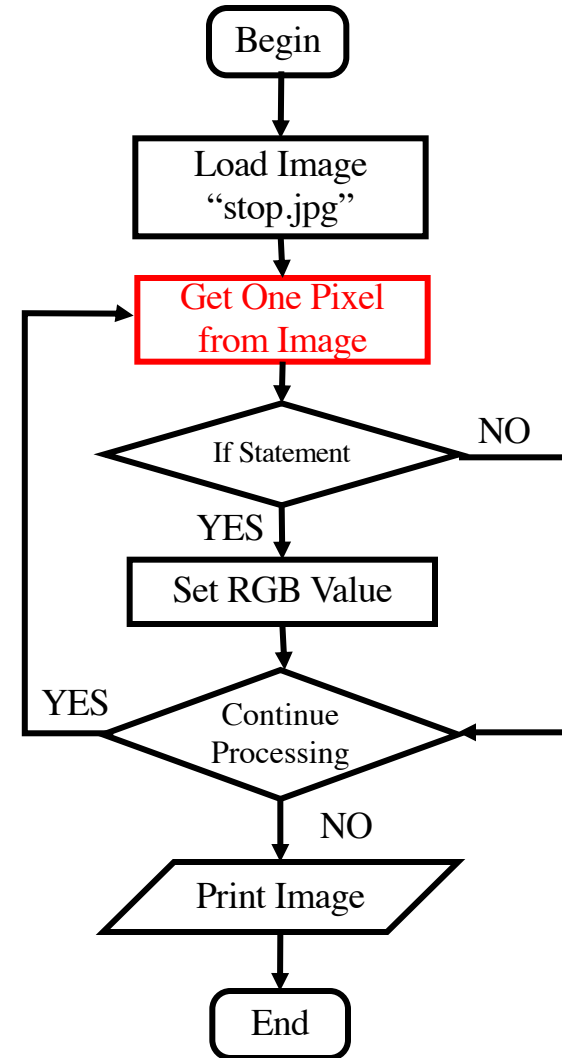
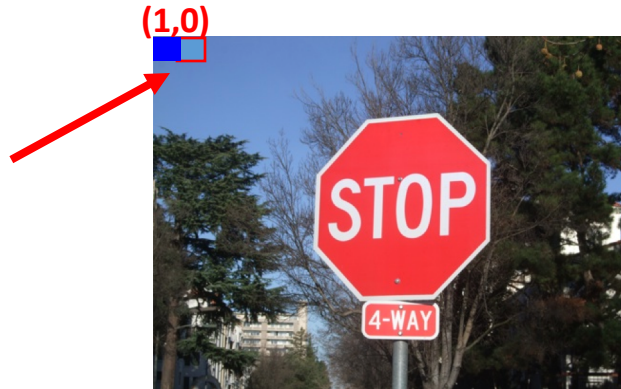
```

image = new SimpleImage("stop.jpg");

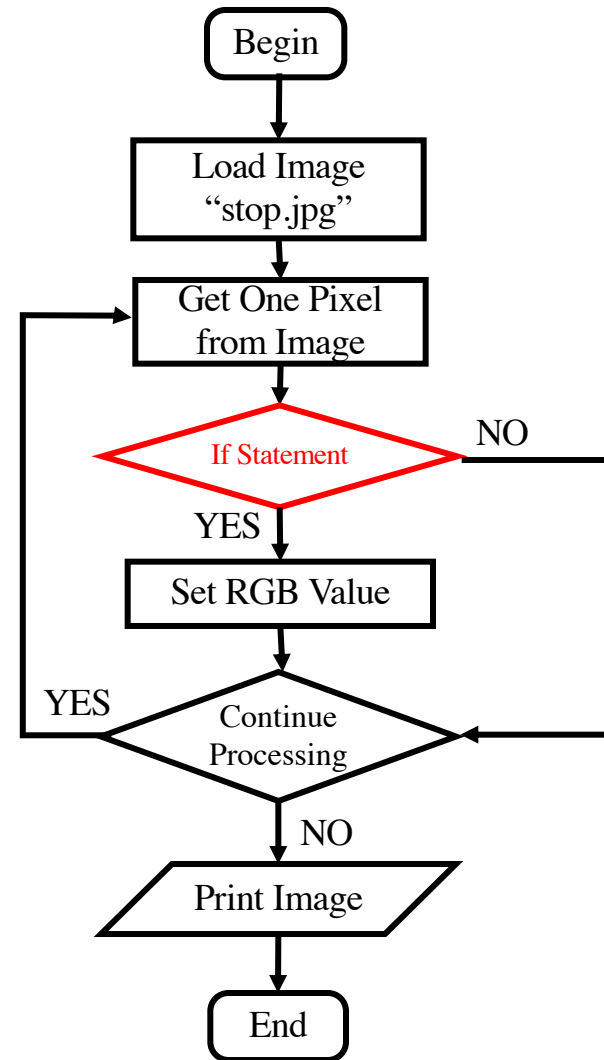
for (pixel : image) {
  if (pixel.getX() < 100) {
    pixel.setRed(255);
    pixel.setGreen(255);
    pixel.setBlue(255);
  }
}

print (image);

```



```
image = new SimpleImage("stop.jpg");  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
print (image);
```



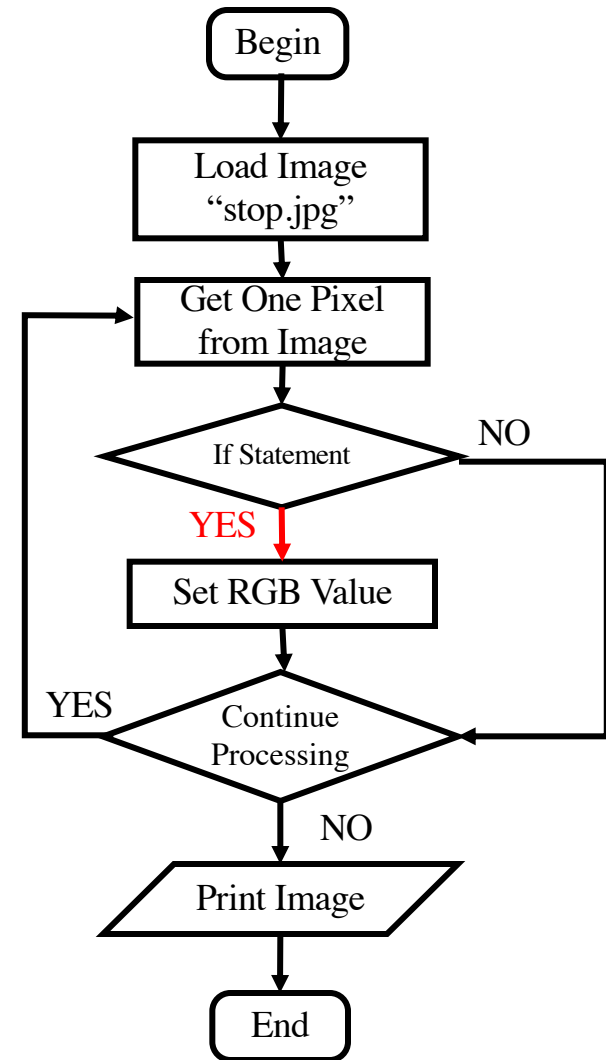
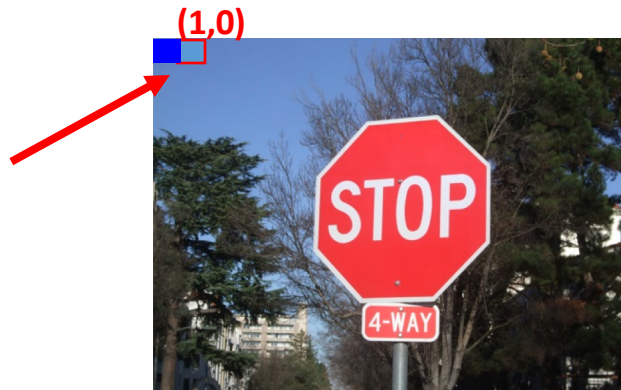
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



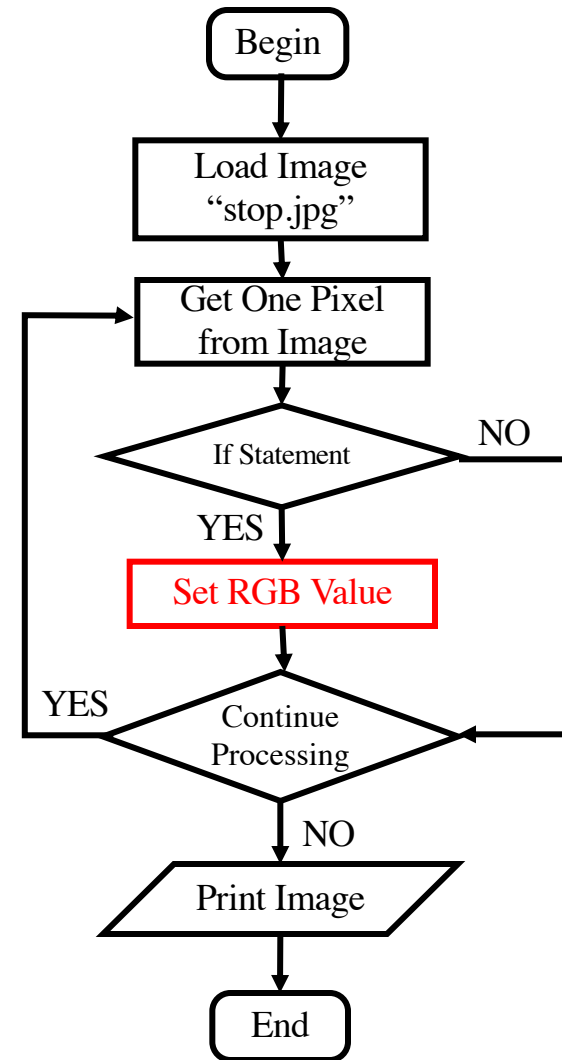
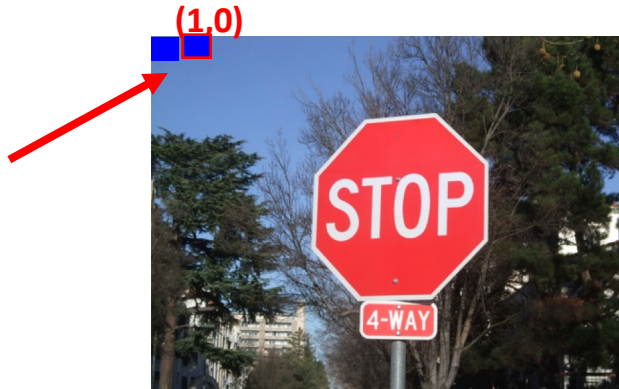
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

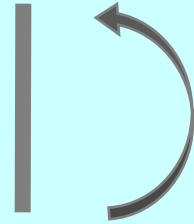
```



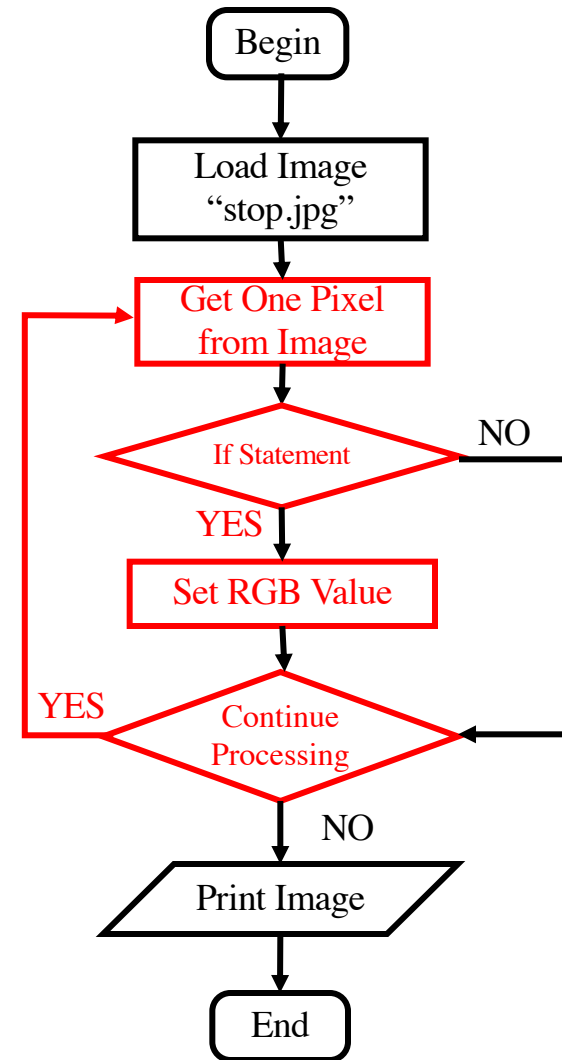

```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}
```

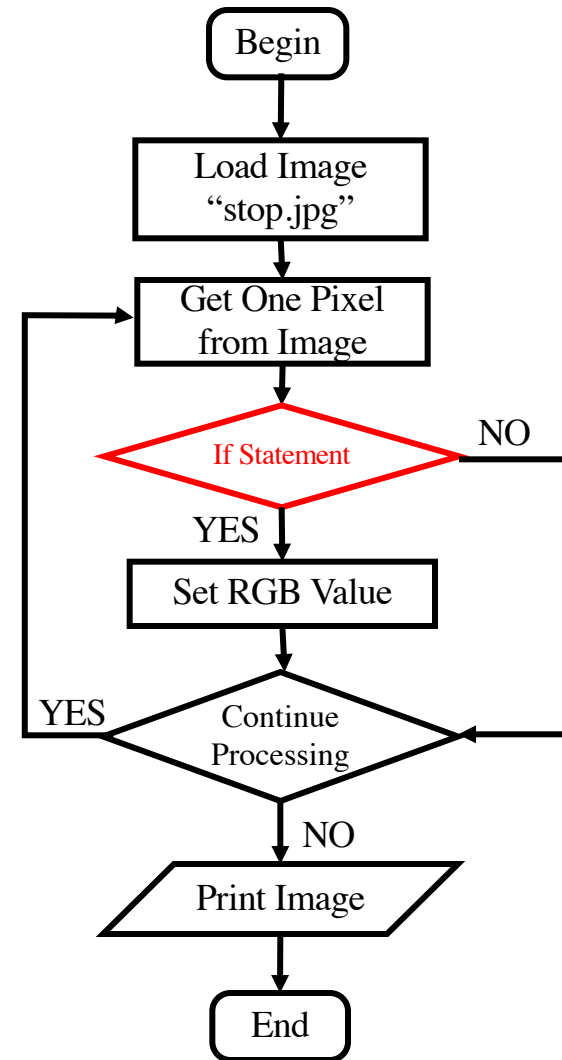
```
print (image);
```



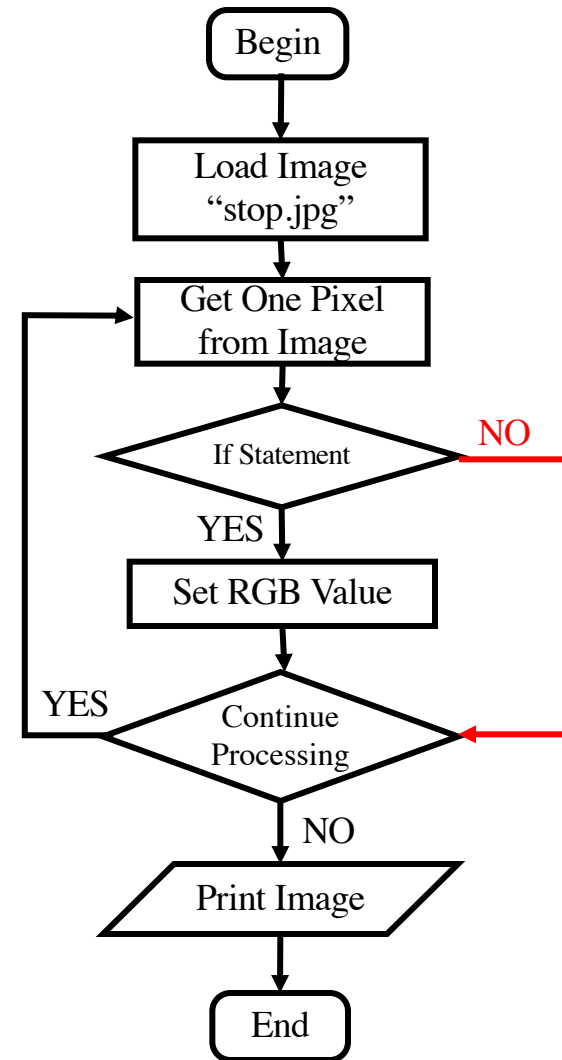
computer
runs it
again and
again,
until to (0, 99)



```
image = new SimpleImage("stop.jpg");  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
print (image);
```



```
image = new SimpleImage("stop.jpg");  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
print (image);
```



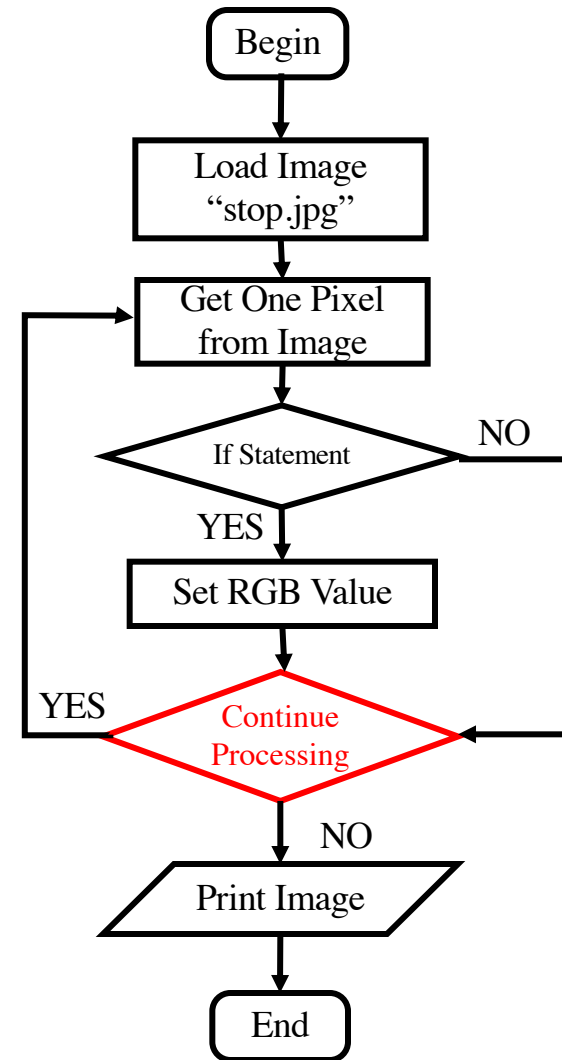
```

image = new SimpleImage("stop.jpg");

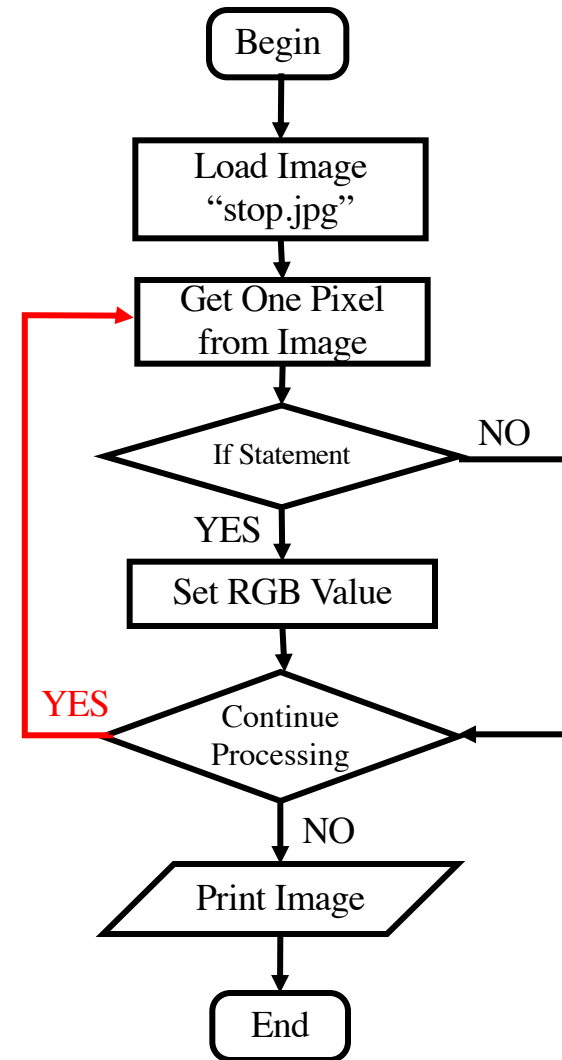
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



```
image = new SimpleImage("stop.jpg");  
  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
  
print (image);
```



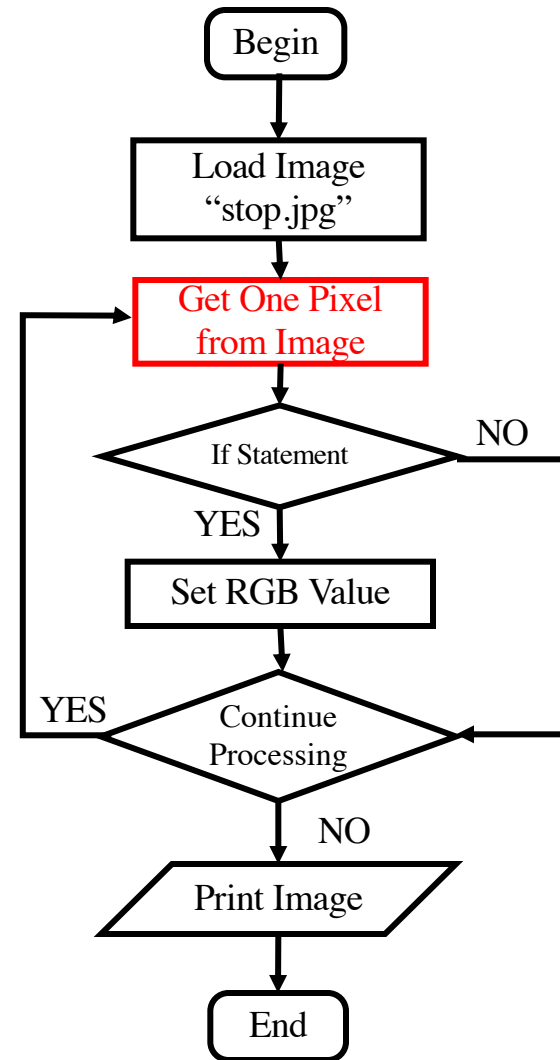
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



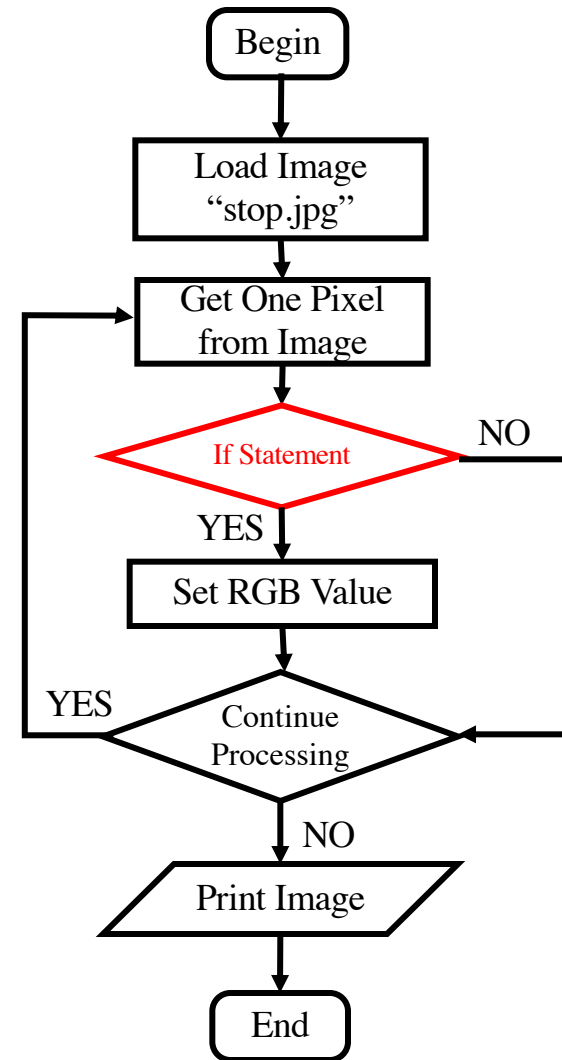
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



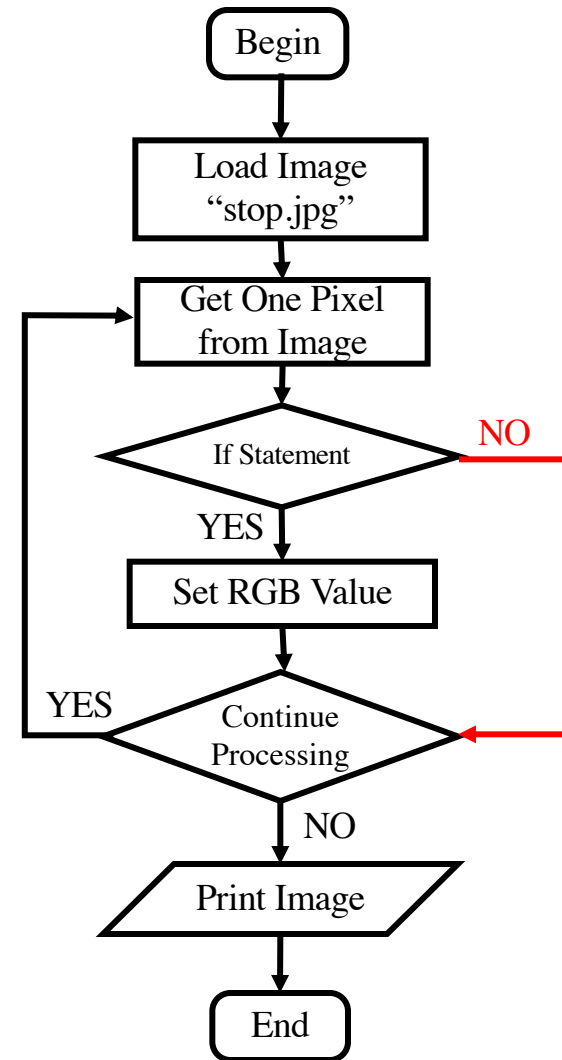
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

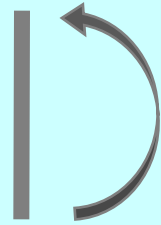
```



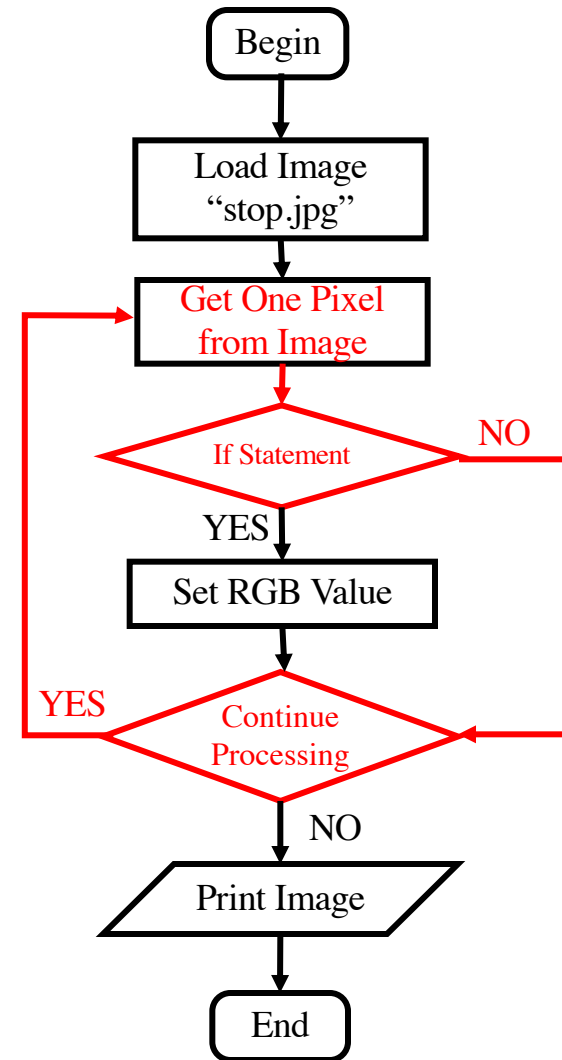

```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}
```

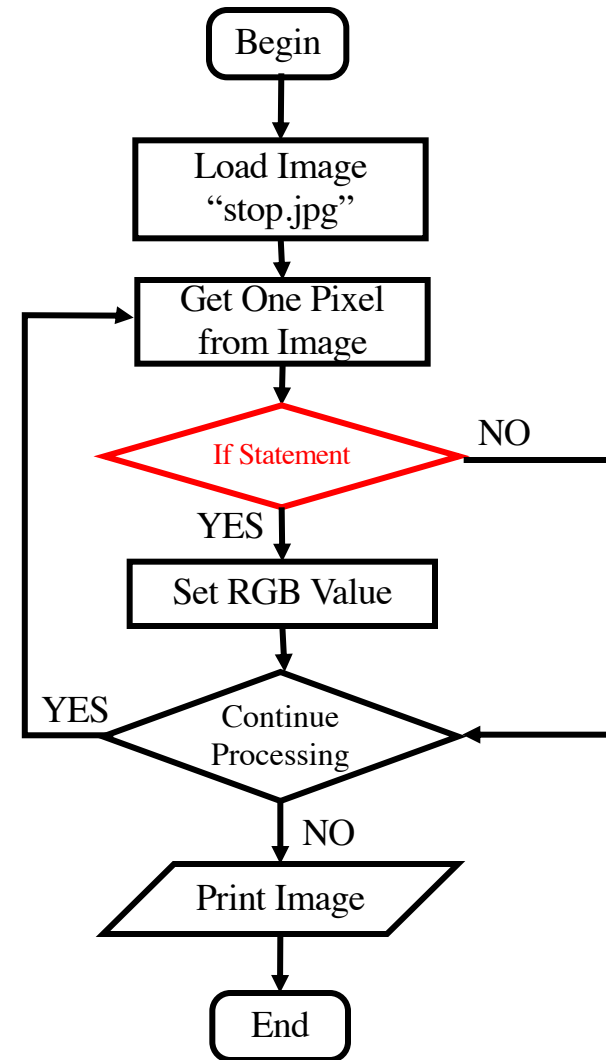
```
print (image);
```



computer
runs it
again and
again,
until to (0, 499)



```
image = new SimpleImage("stop.jpg");  
  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
  
print (image);
```



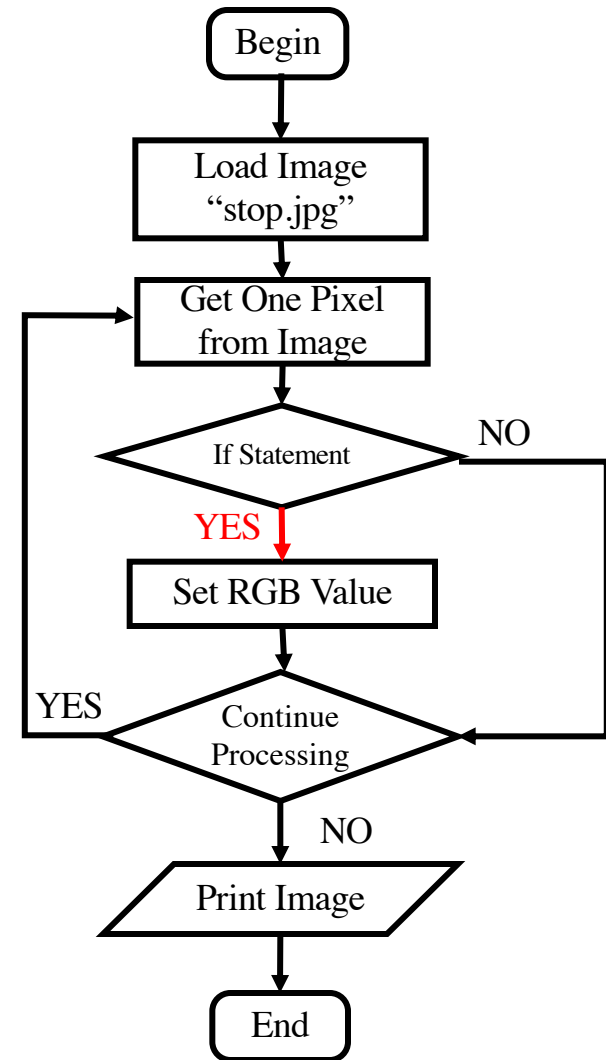
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



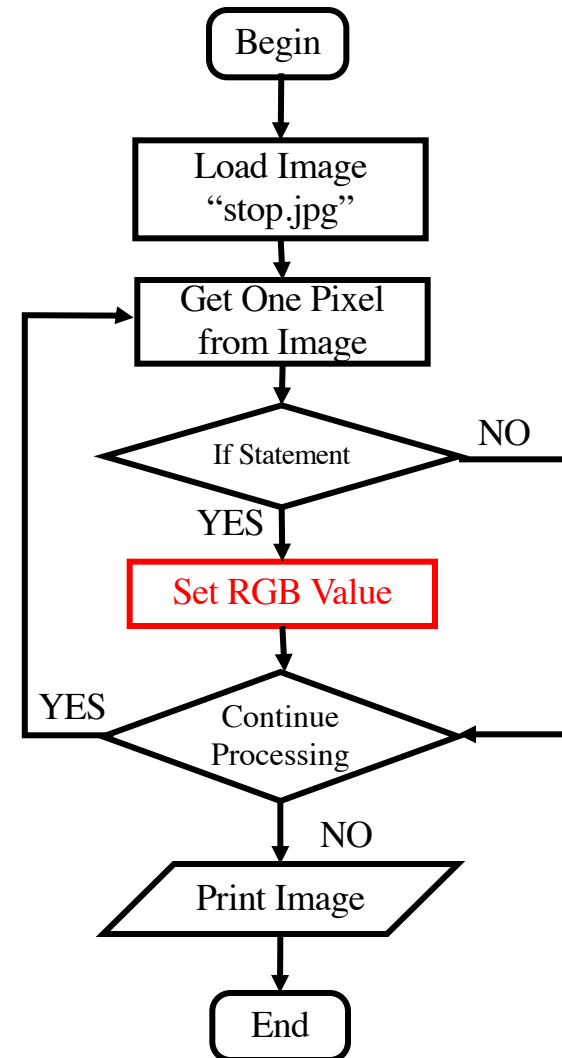
```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

print (image);

```



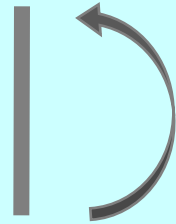


```

image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

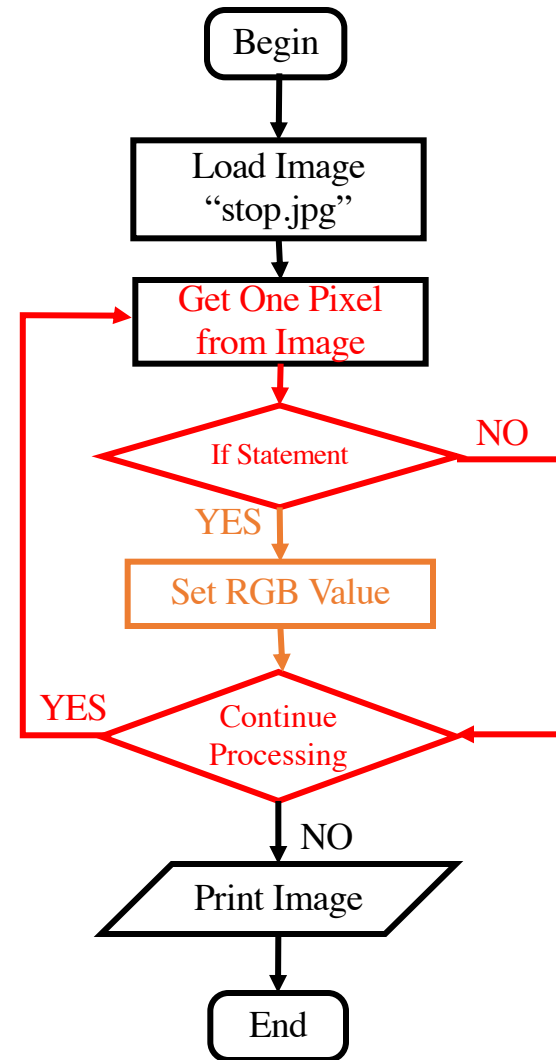
print (image);
    
```



computer runs it again and again, until to (319, 499)



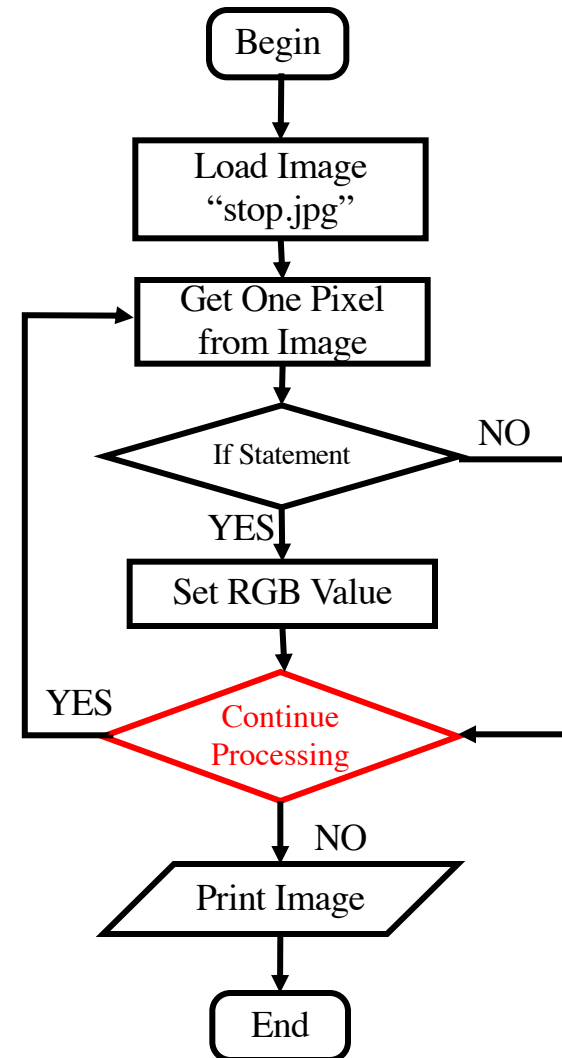
(319,499)



```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}
```

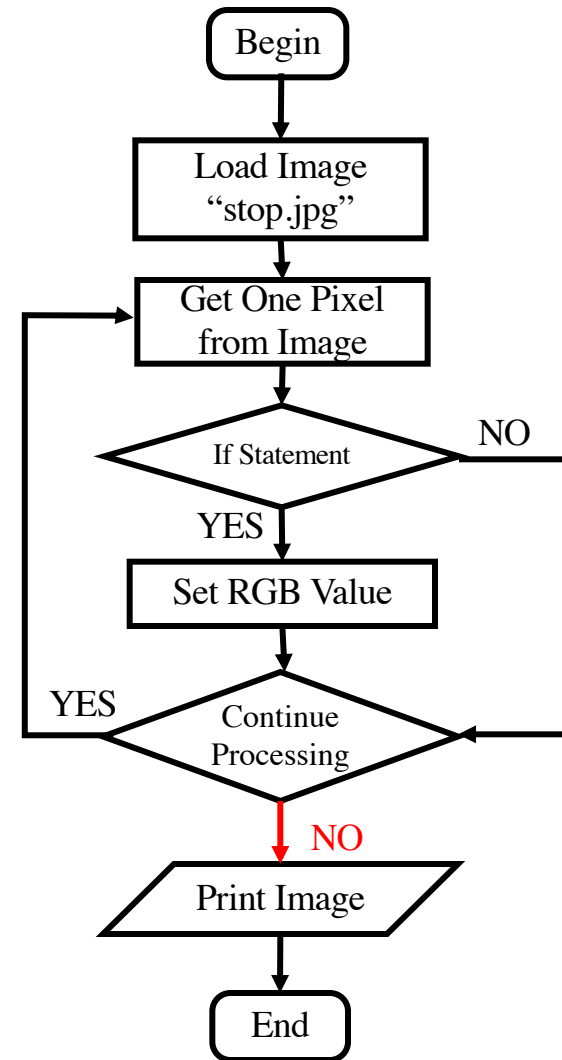
```
print (image);
```



```
image = new SimpleImage("stop.jpg");
```

```
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}
```

```
print (image);
```



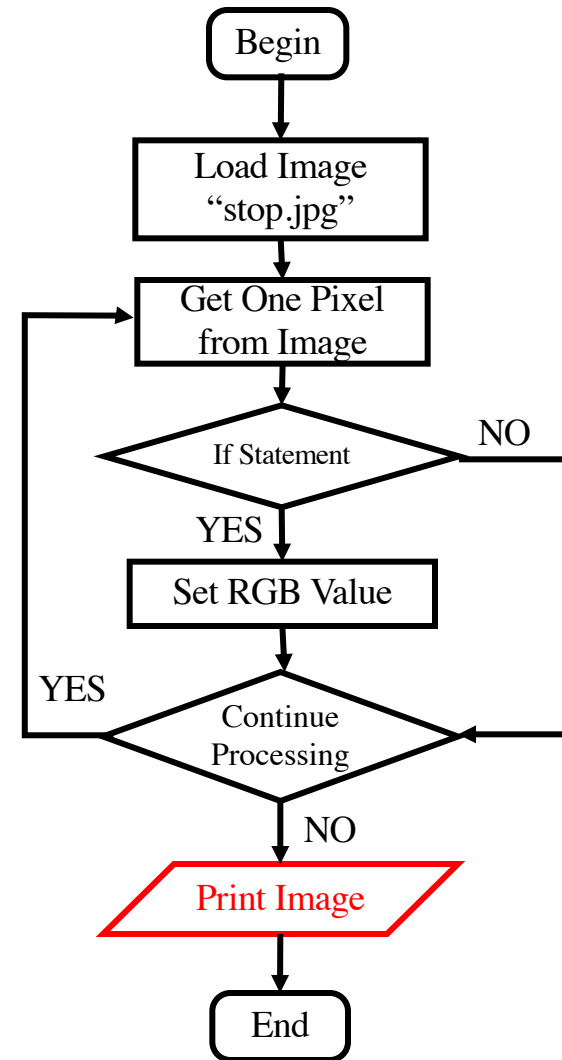
```

image = new SimpleImage("stop.jpg");

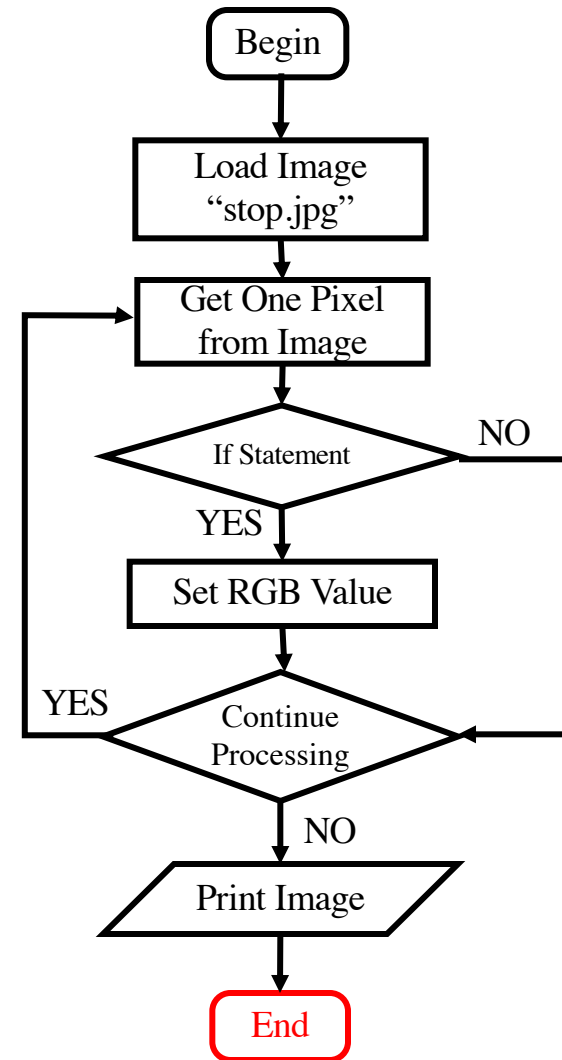
for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(255);
        pixel.setGreen(255);
        pixel.setBlue(255);
    }
}

```

```
print (image);
```




```
image = new SimpleImage("stop.jpg");  
  
for (pixel : image) {  
    if (pixel.getX() < 100) {  
        pixel.setRed(255);  
        pixel.setGreen(255);  
        pixel.setBlue(255);  
    }  
}  
  
print (image);
```





If statement :

- using $<$ (less than) or $>$ (greater than) in test
- Key pattern: the loop hits every x,y . Only some of them pass the test.

Type:

- a. if with location:

.e.g. `(pixel.getX() < 150)`
or `(pixel.getY() > 100)`

- b. if with color:

.e.g. `(pixel.getRed() > 160)`
or `(pixel.getRed() > avg * 1.1)`

where `avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue())/3;`

Example 1

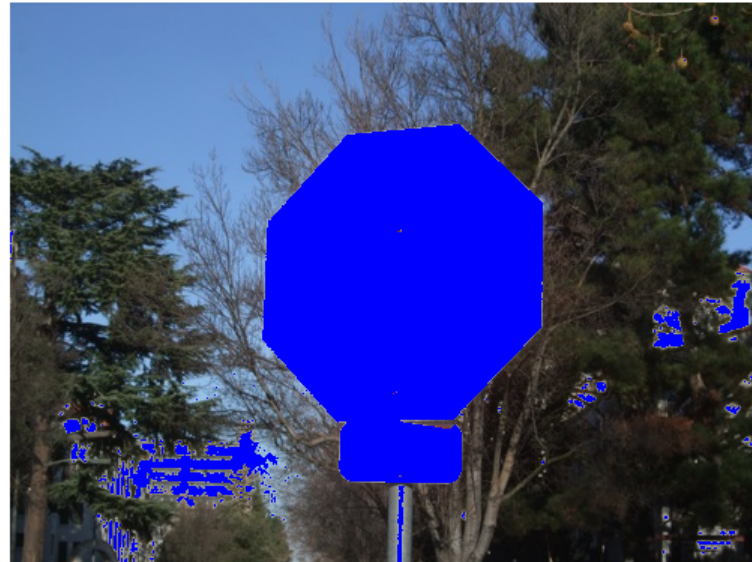
```
image = new SimpleImage("stop.jpg");
for (pixel: image) {
    if ( pixel.getX() > 355) {
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(0);
    }
}
print(image);
```





Example 2

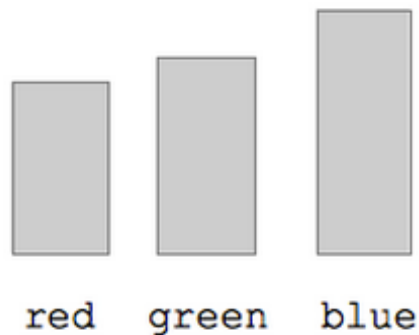
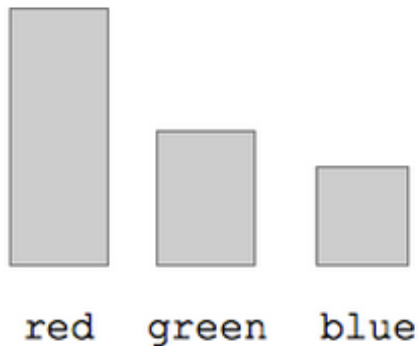
```
image = new SimpleImage("stop.jpg");
for (pixel: image) {
    if (pixel.getRed() > 160) {
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(255);
    }
}
print(image);
```





Color Bars

Suppose we have three pixels. Each pixel has the familiar red/green/blue values. Suppose these three values are graphed as bar graphs like this:





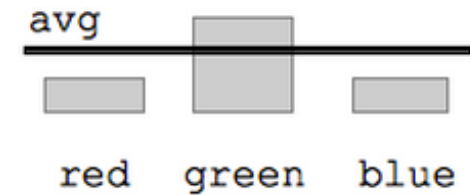
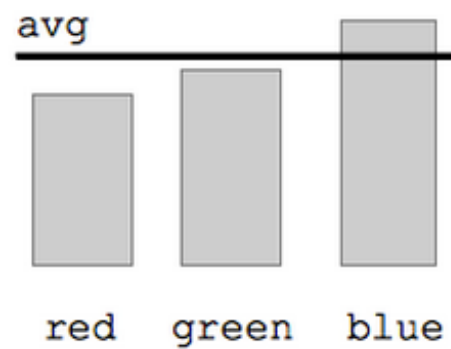
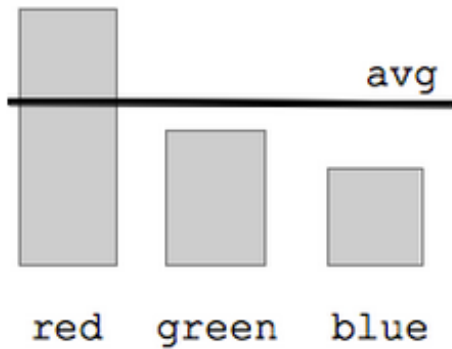
Recall Average of Pixel Color

Recall this code which, inside a loop, computes the average of the red, blue, and green values and stores that number in a variable "avg"

```
avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue())/3;
```

Color Bars With Average

Here are the three pixels again, but now the **avg** value is show as a line drawn across the bars.



Example 3: Color Avg Test

```
image = new SimpleImage("stop.jpg");
for (pixel: image) {
    avg = (pixel.getRed() + pixel.getGreen() +
pixel.getBlue())/3;

    if (pixel.getRed() > avg * 1.1) {
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(255);
    }
}
print(image);
```



Example 4: Color Avg Test

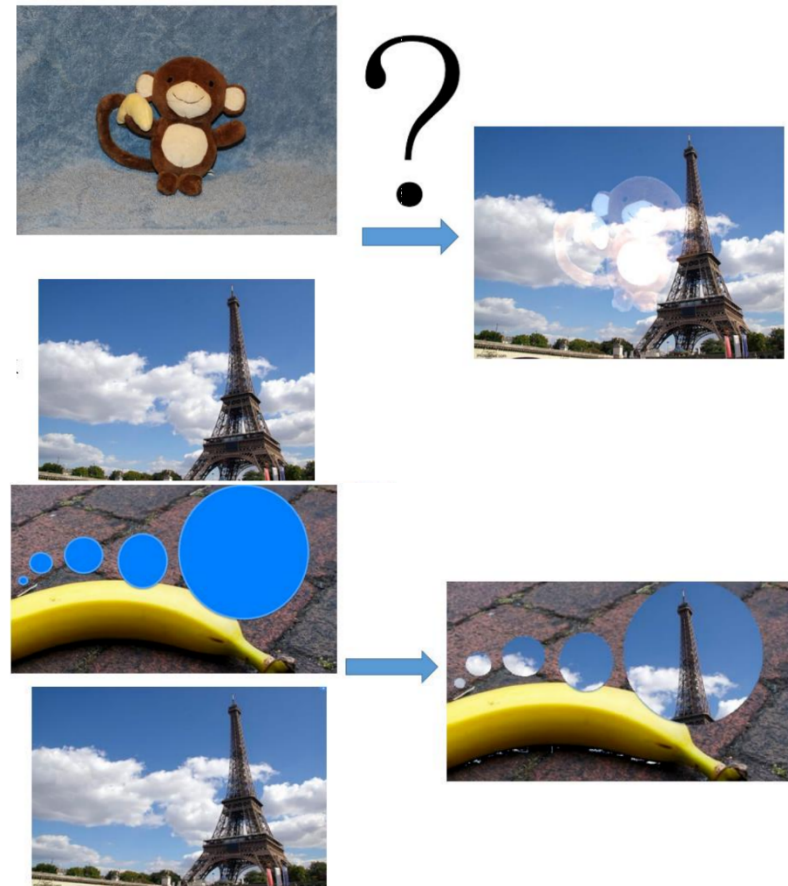
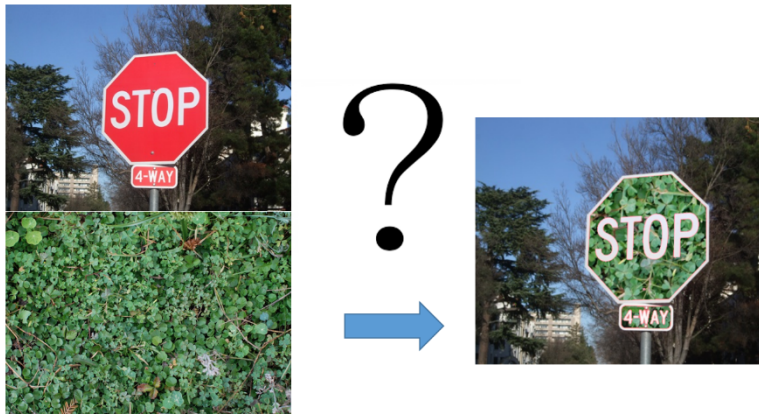
```
image = new SimpleImage("curb.jpg");  
for (pixel: image) {  
    avg = (pixel.getRed() + pixel.getGreen() +  
    pixel.getBlue())/3;
```

```
    if (pixel.getRed() > avg * 1.1) {  
        pixel.setRed(120);  
        pixel.setGreen(120);  
        pixel.setBlue(120);  
    }  
}  
print(image);
```



Flipped Classroom (翻转课堂)

- **Subject: Special Effect on Image**
 - Bluescreen Special Effect (蓝屏特效)
 - Image Blend (图像融合)





Flipped Classroom (翻转课堂)

- **Subject:** Special Effect on Image
 - [Bluescreen Special Effect \(蓝屏特效\)](#)
 - [Image Blend \(图像融合\)](#)
- **Requirement:**
 - Submit your PPT slides before **October 11th**;
 - 4 best groups will be selected for presentation;
 - Presentation time: no more than 15 minutes;
 - Try your best, and show your bests.

Bluescreen Special Effect(蓝屏特效)



How can we change the red part of sign to look like leaves?

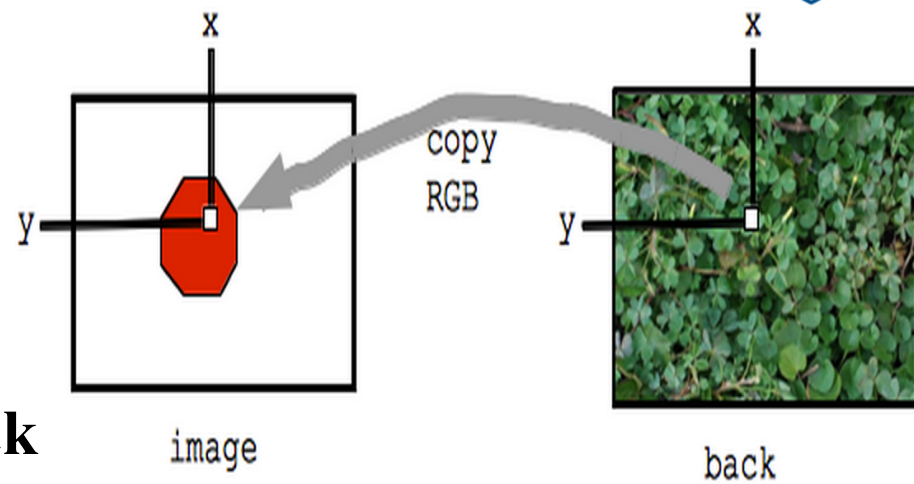


BlueScreen Special Effect(Chroma Key)

- "Bluescreen" video special effect
- Video is just a series of still images, 20-60 per second
- Have some background image, e.g. leaves.jpg
- Detect colored area in an image, e.g. red in stop.jpg
- For colored area, substitute in pixels from background image



Bluescreen algorithm



Step1: Two images: **image** and **back**

Step2: Detect, say, red pixels in **image**

Step3: For each red pixel:

- 1) Get the x,y in **image**
- 2) Consider the pixel at the same x,y in **back**
- 3) Copy over the red, green, and blue values from **back** to **image**

Result: Red areas, substitute in areas from back image
Adjacent pixels maintained



Two pictures



Image(The stop.jpg image)

Back(The leaves.jpg image)



Bluescreen Example Code

```
image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() + pixel.getGreen() +
pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
```



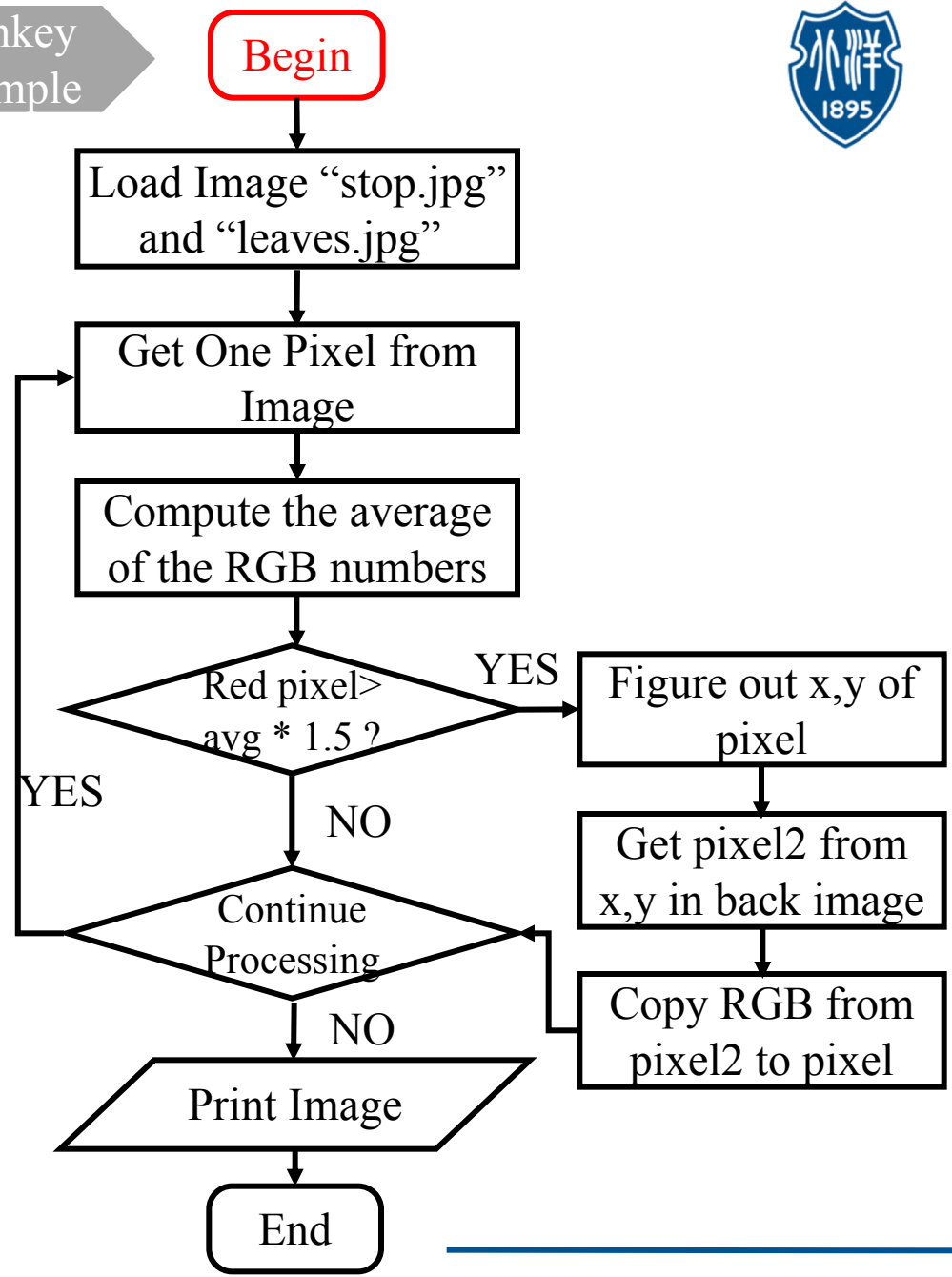


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



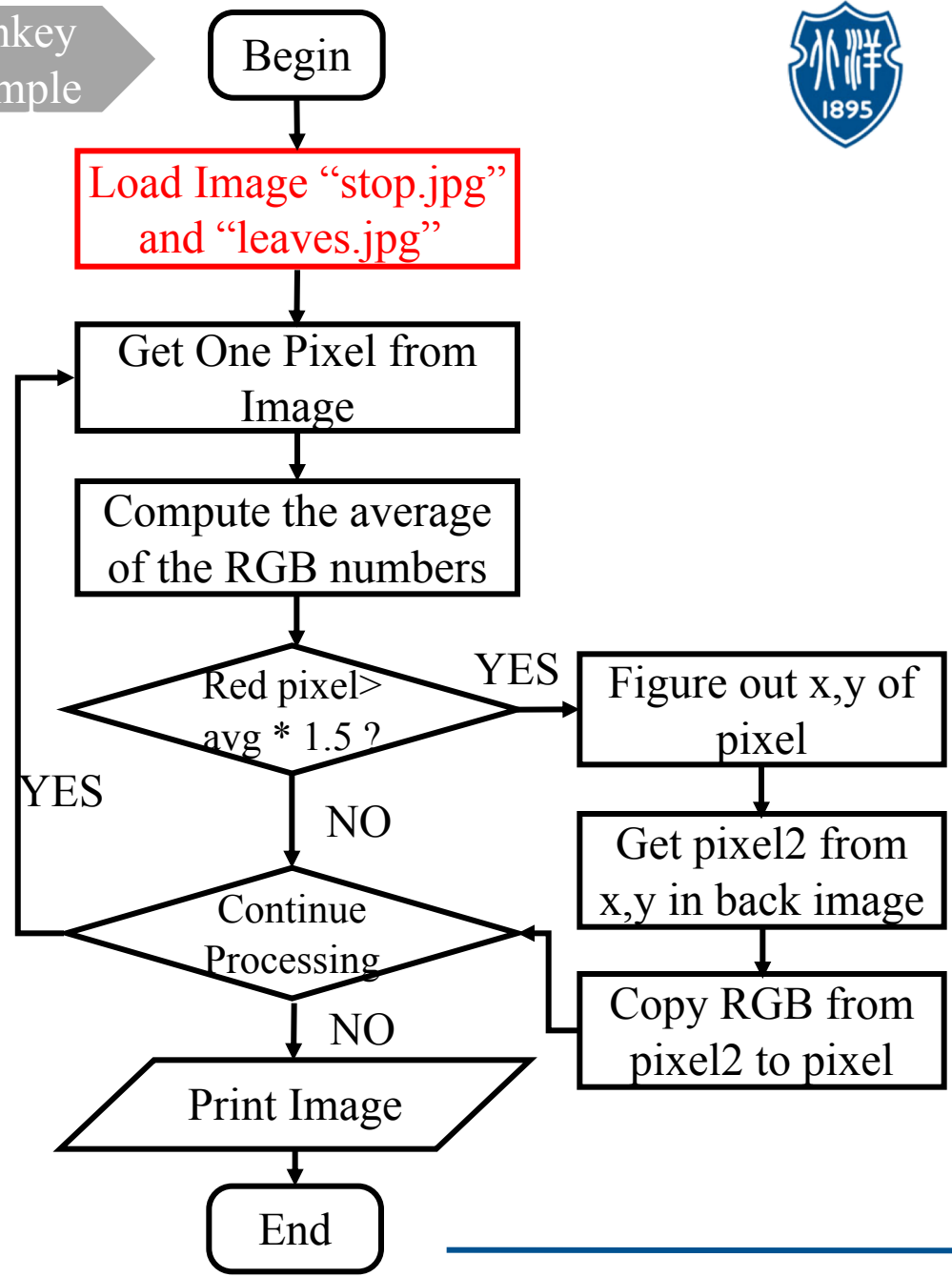


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



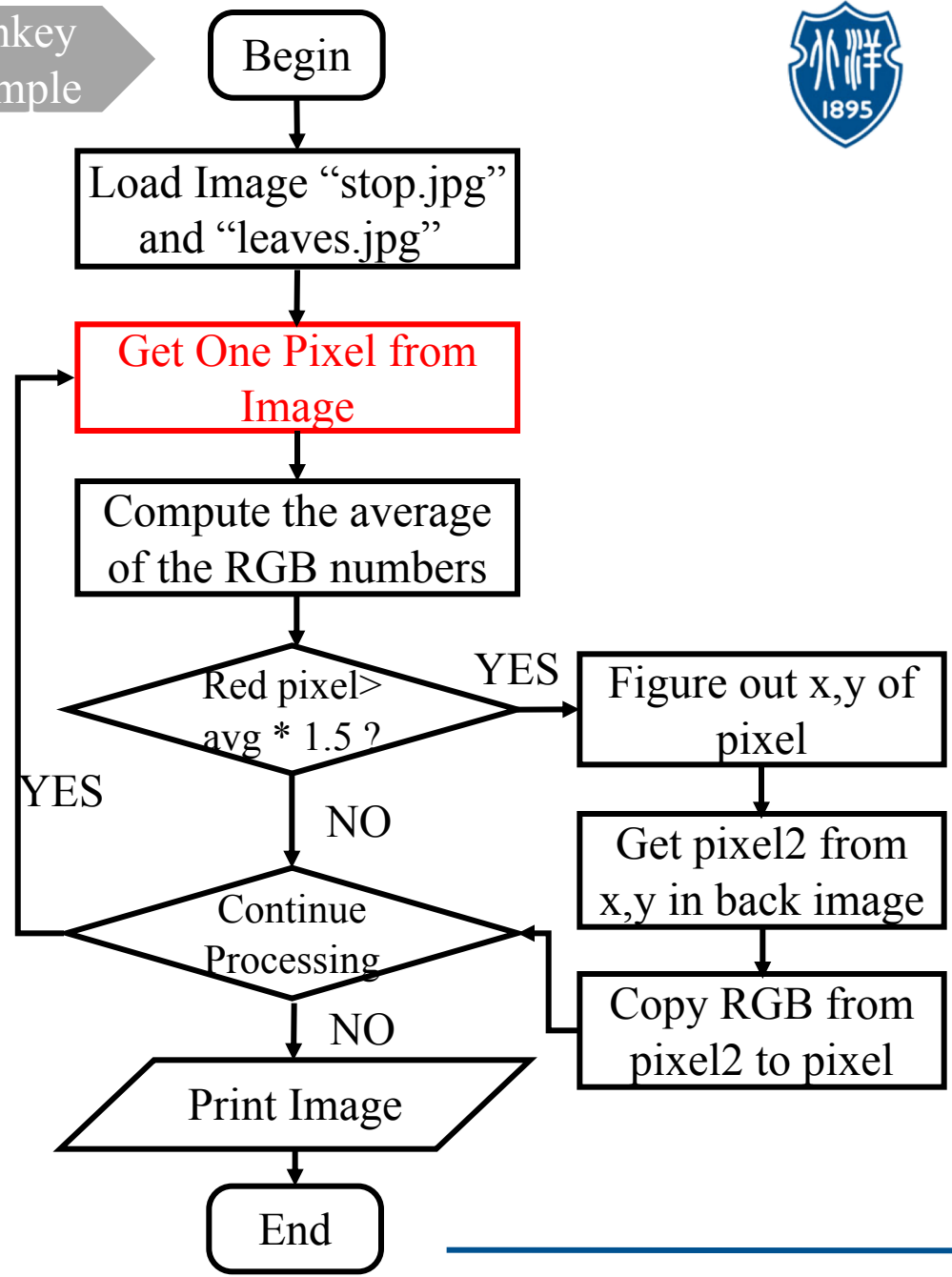


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



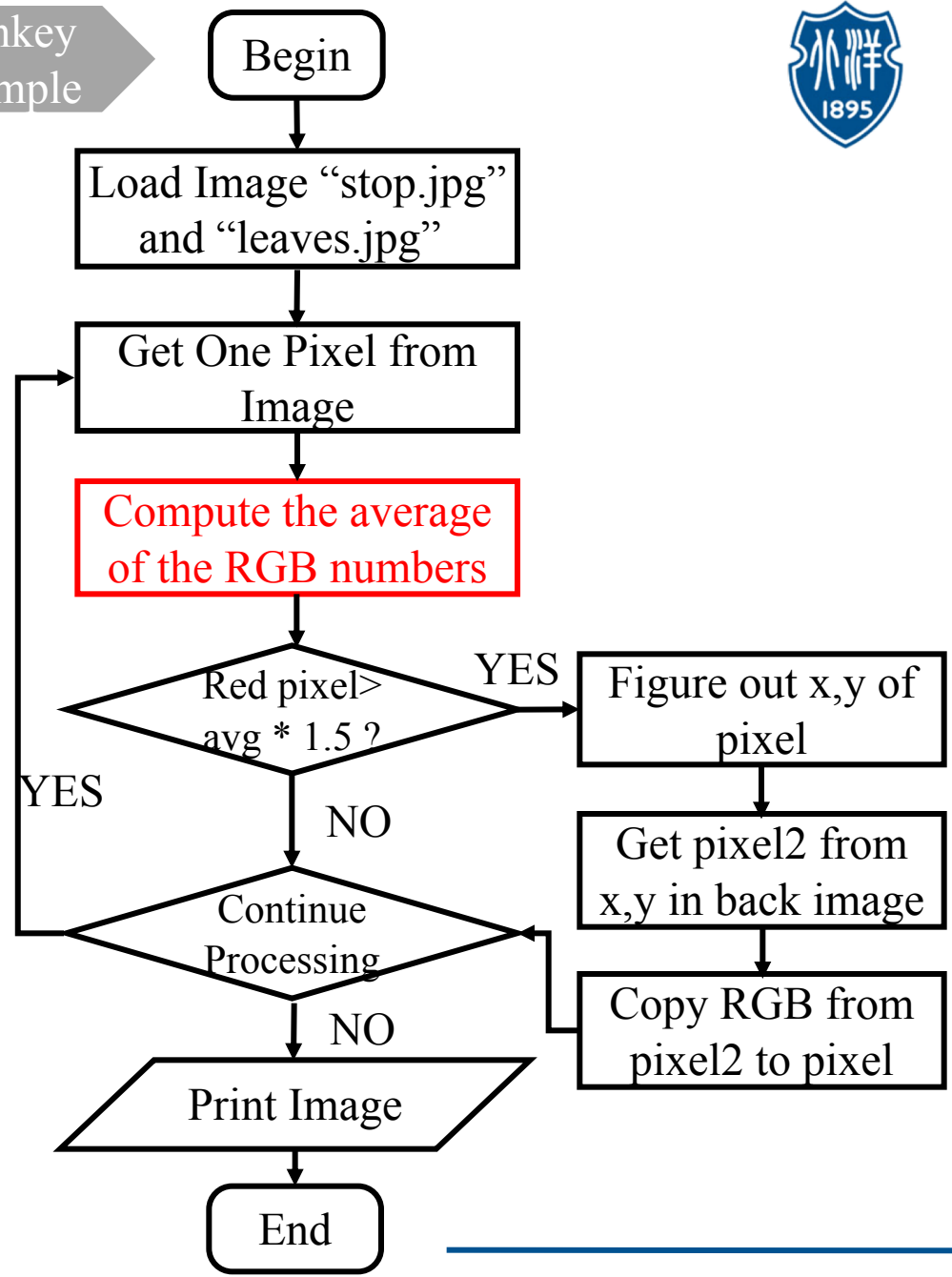


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



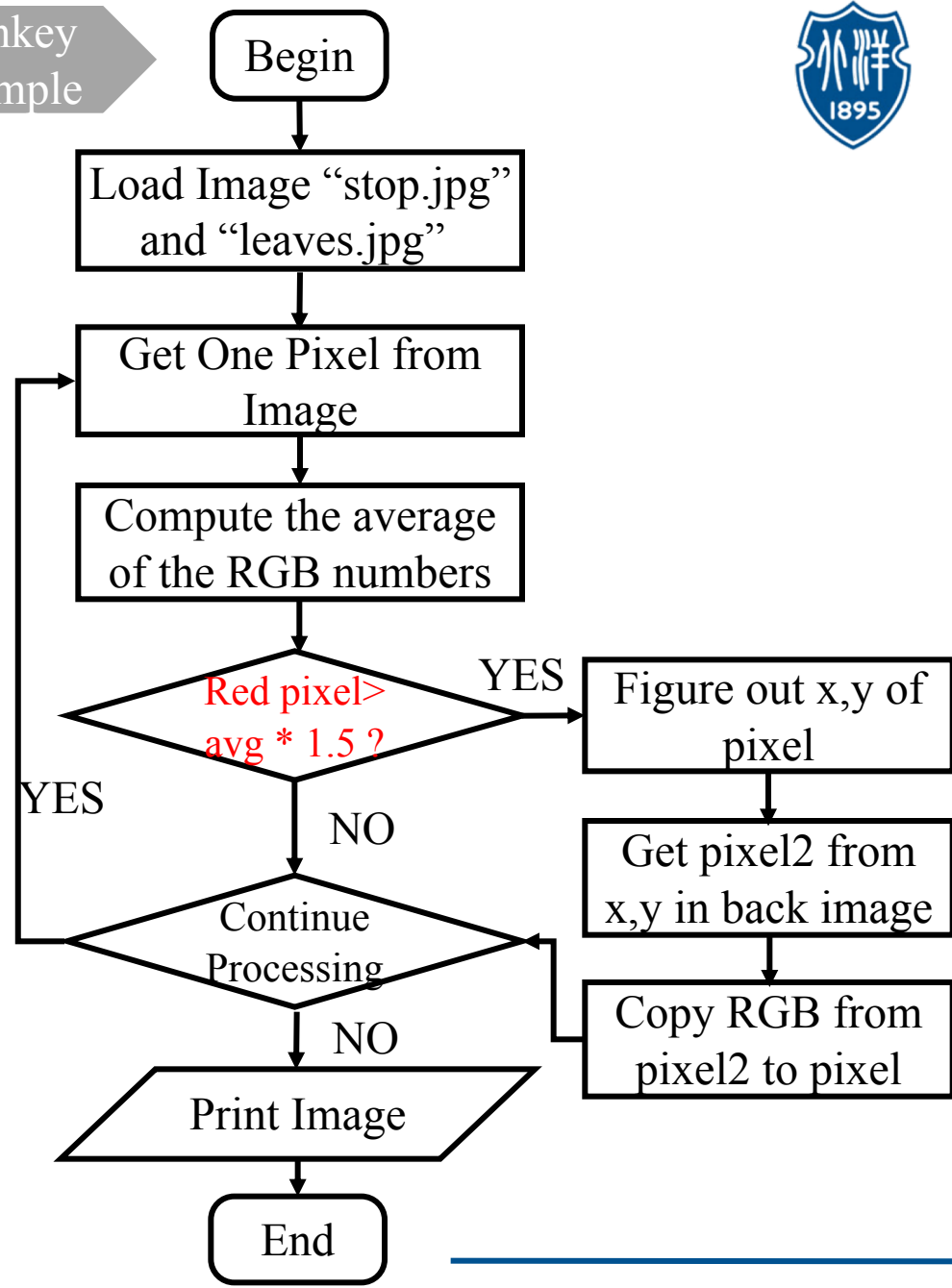


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



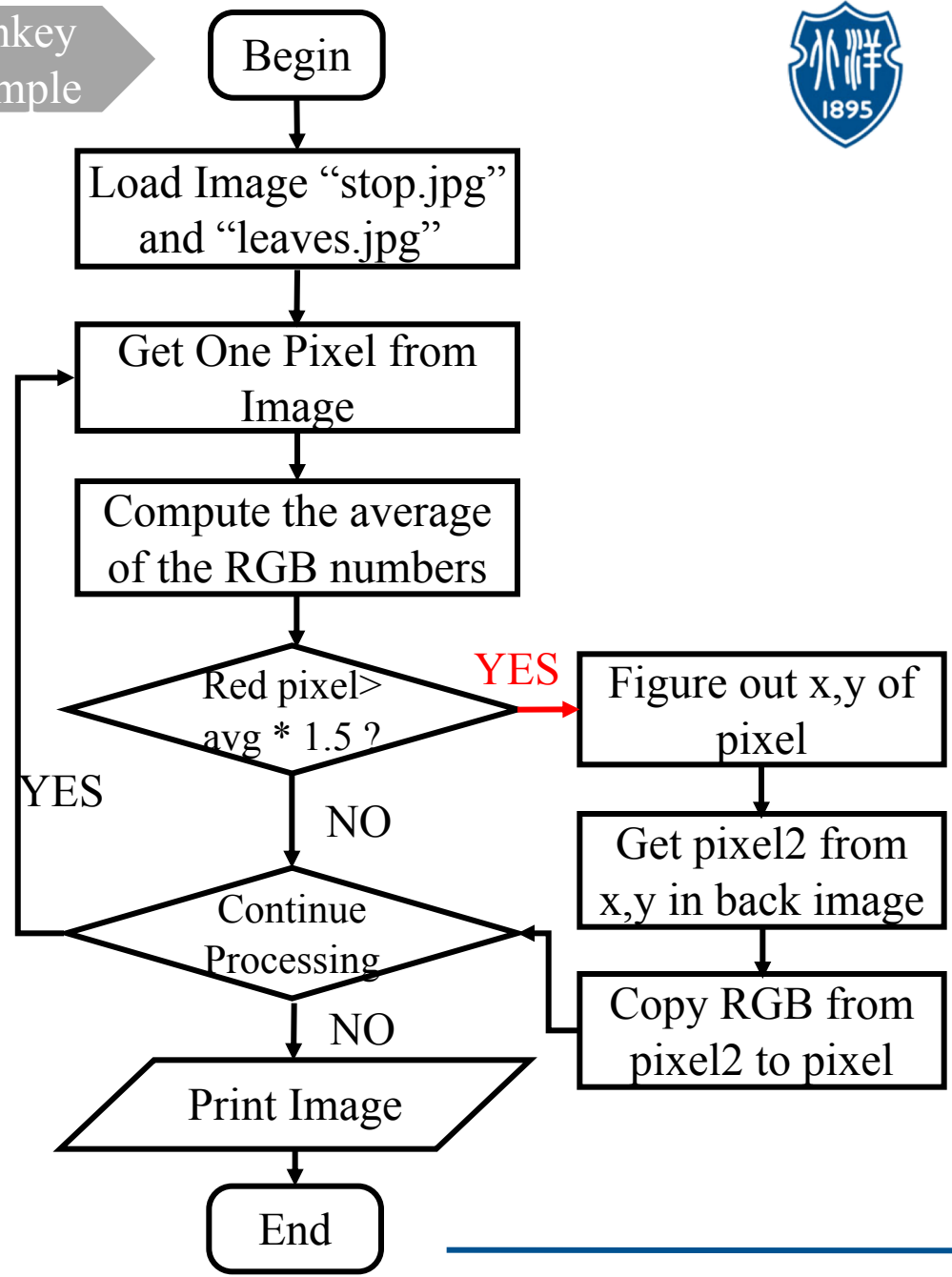


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



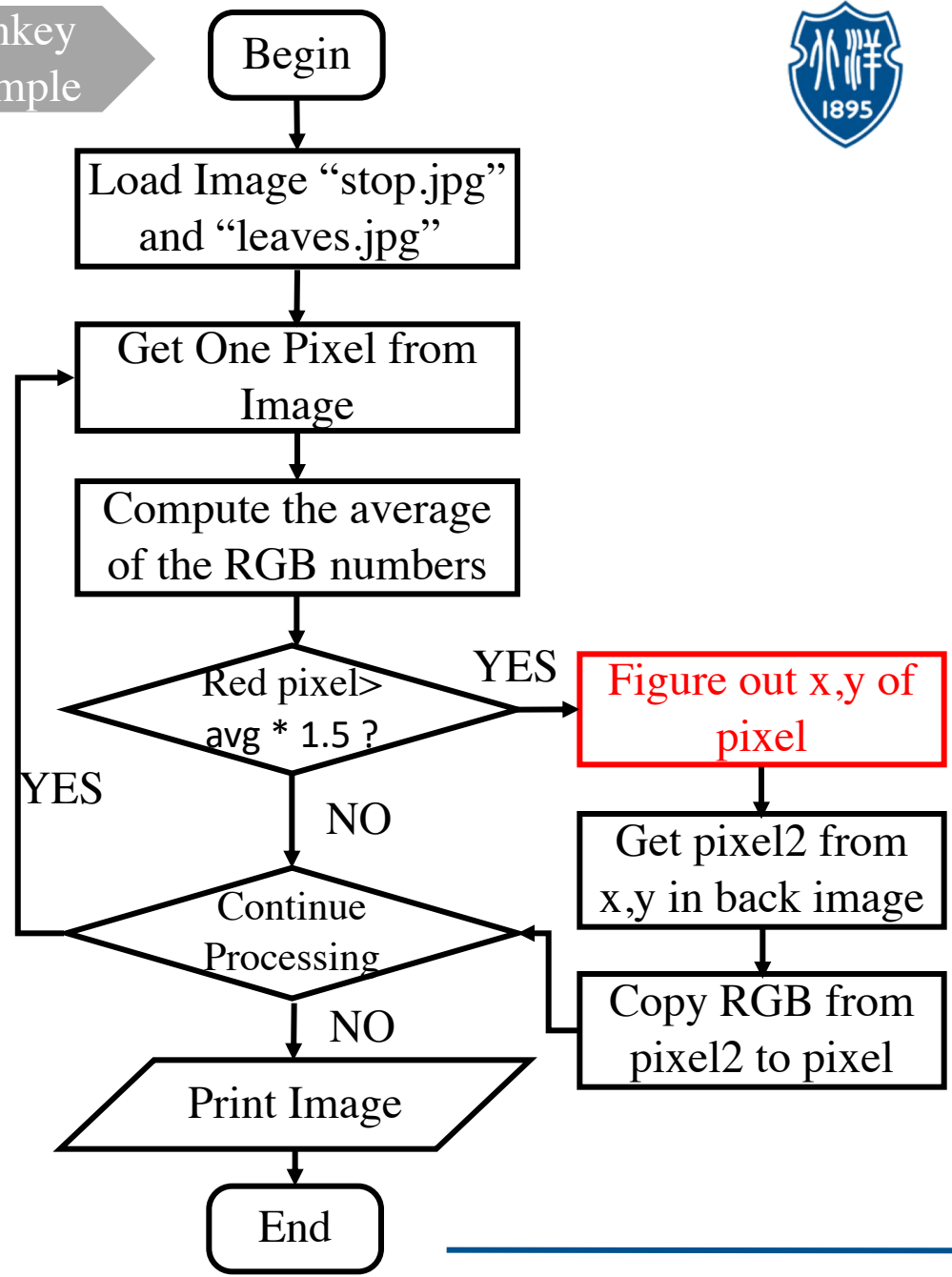


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
    pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



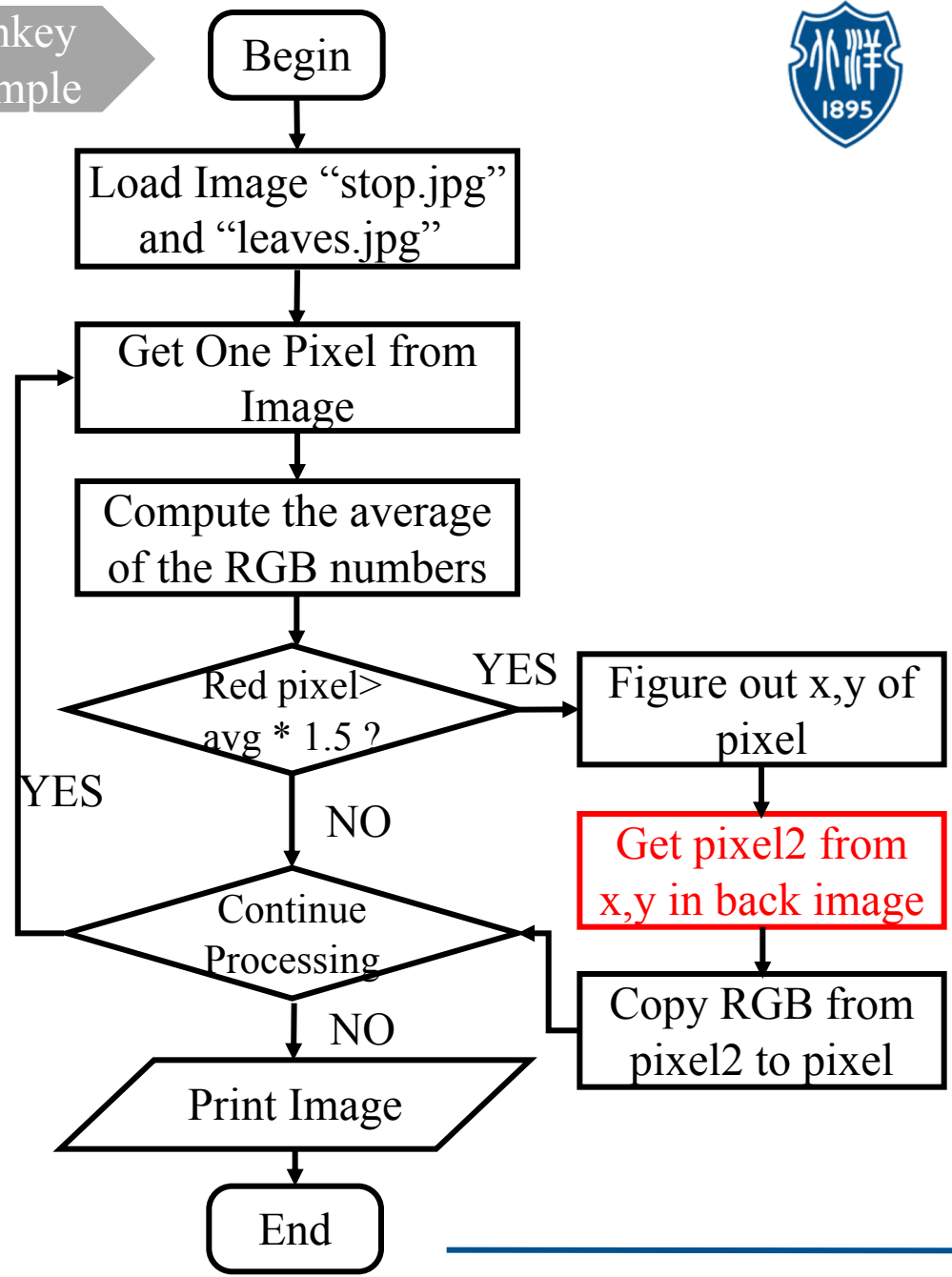


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



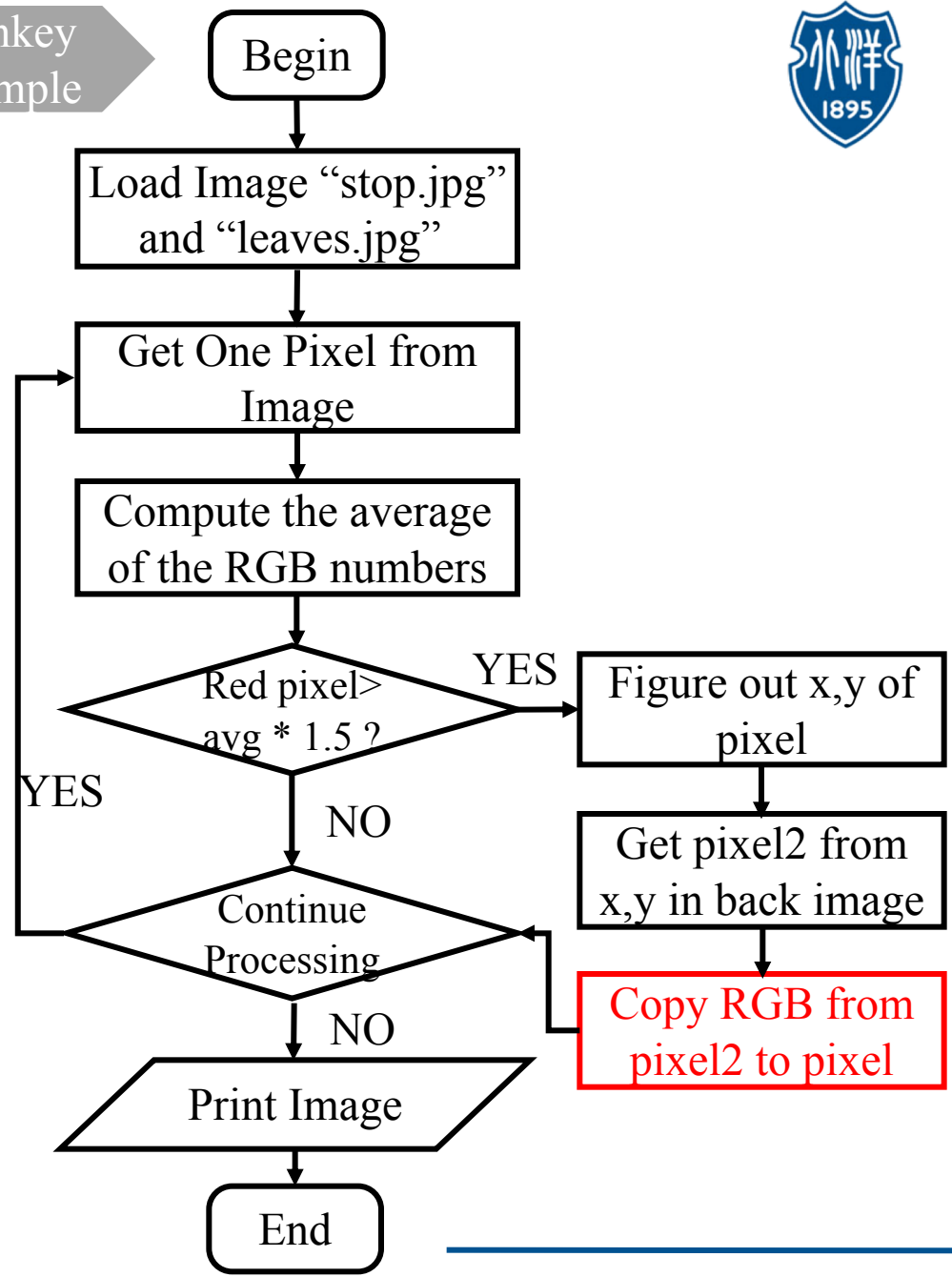


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



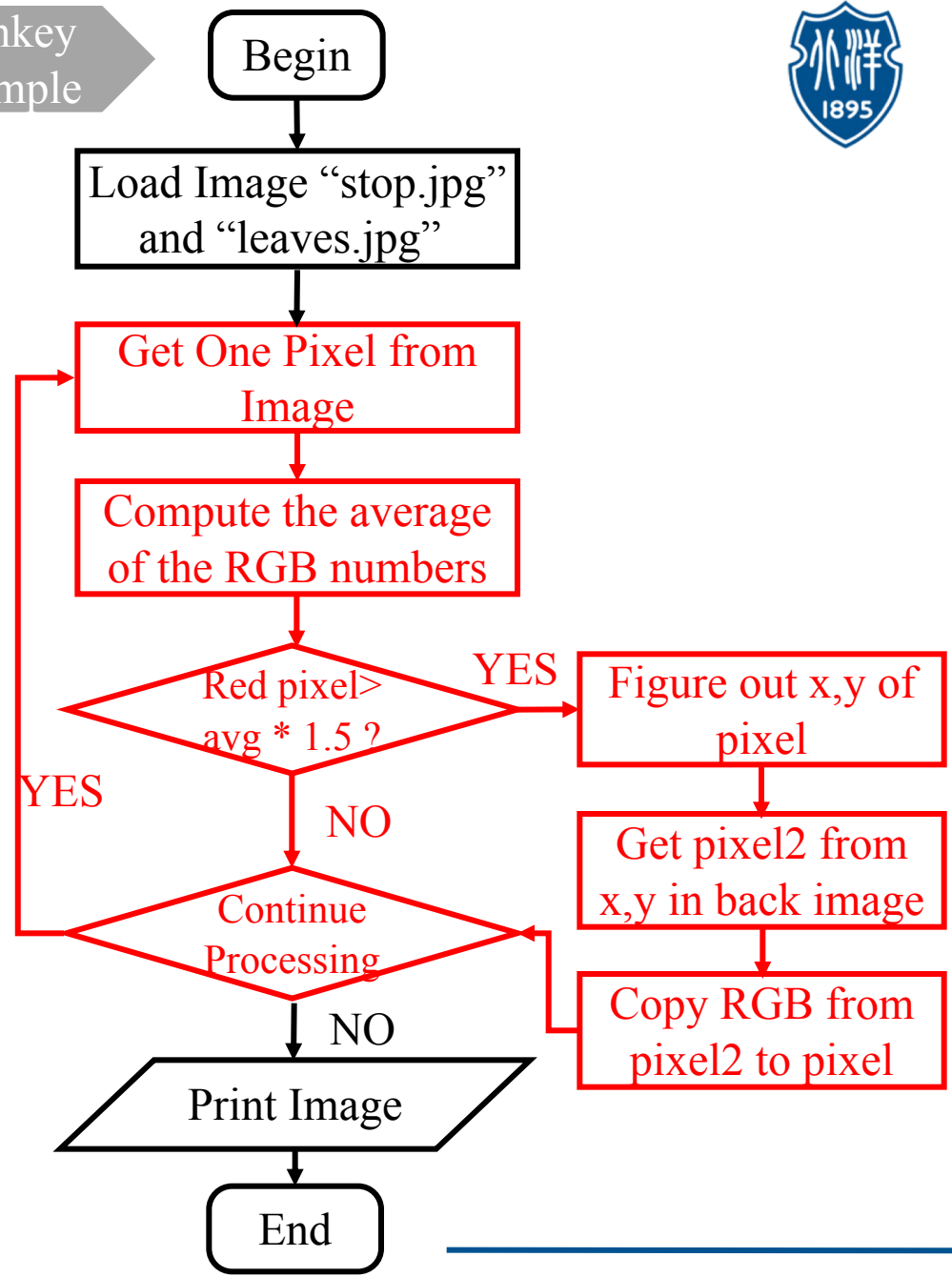


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



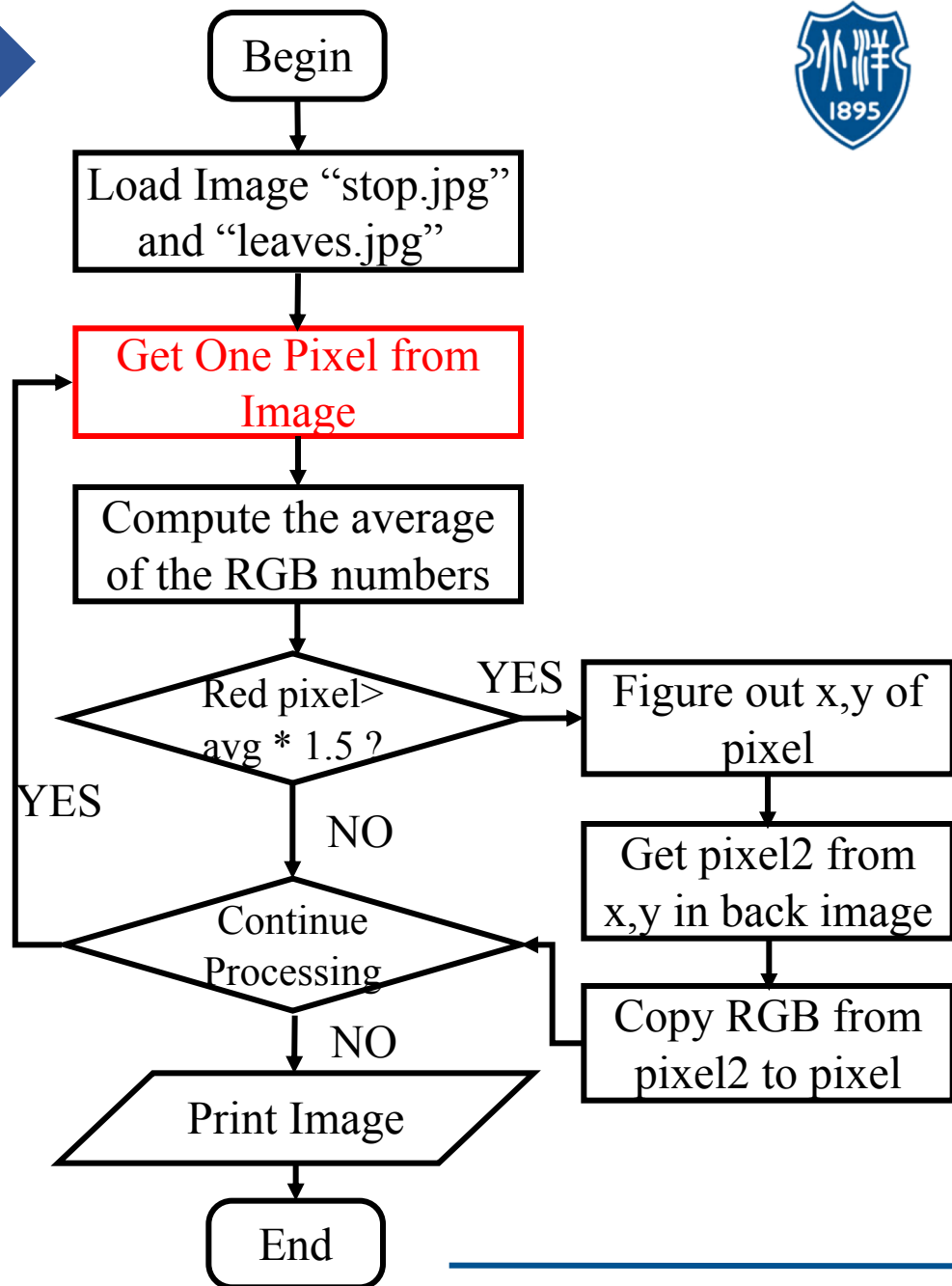


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



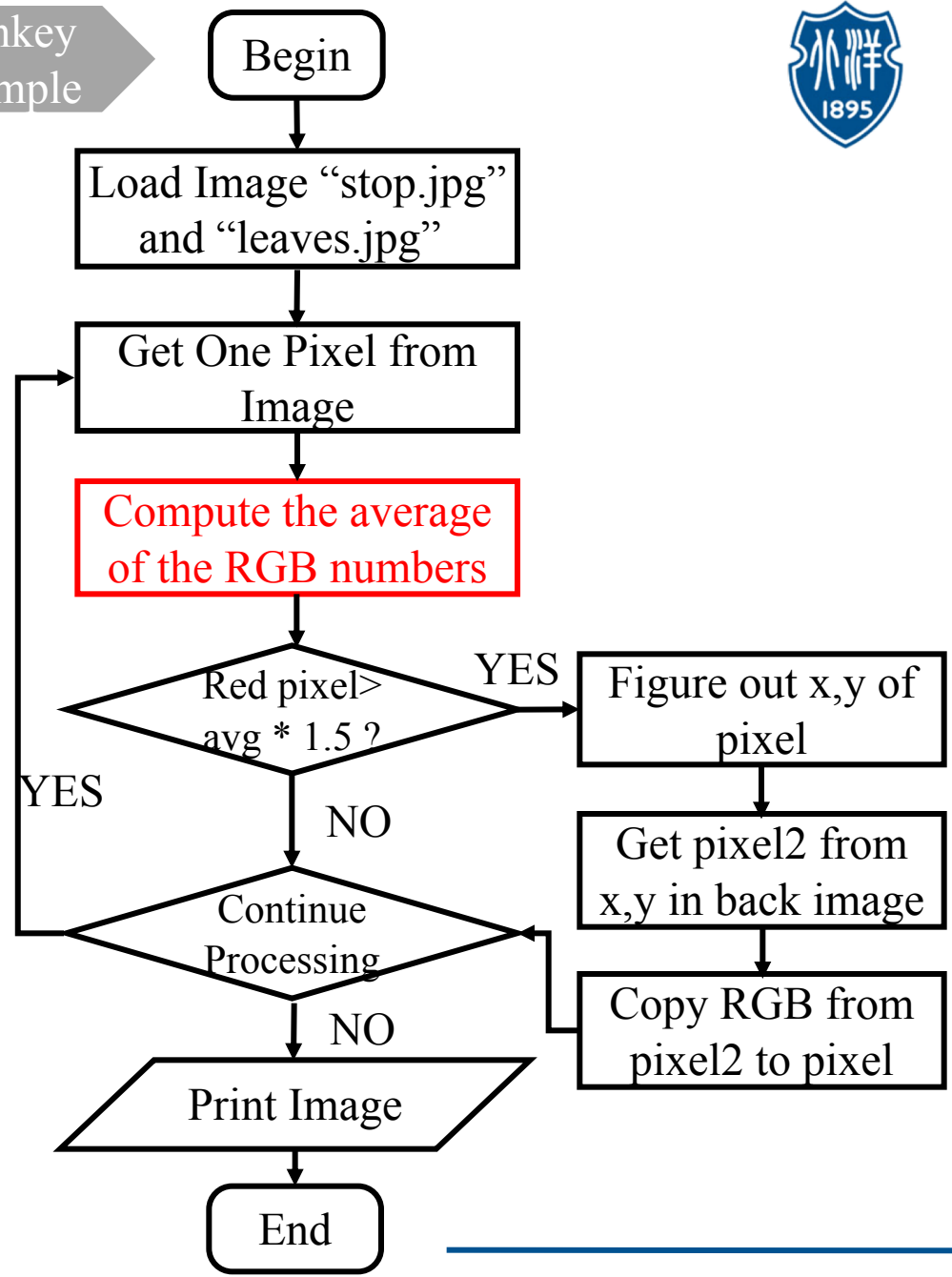


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



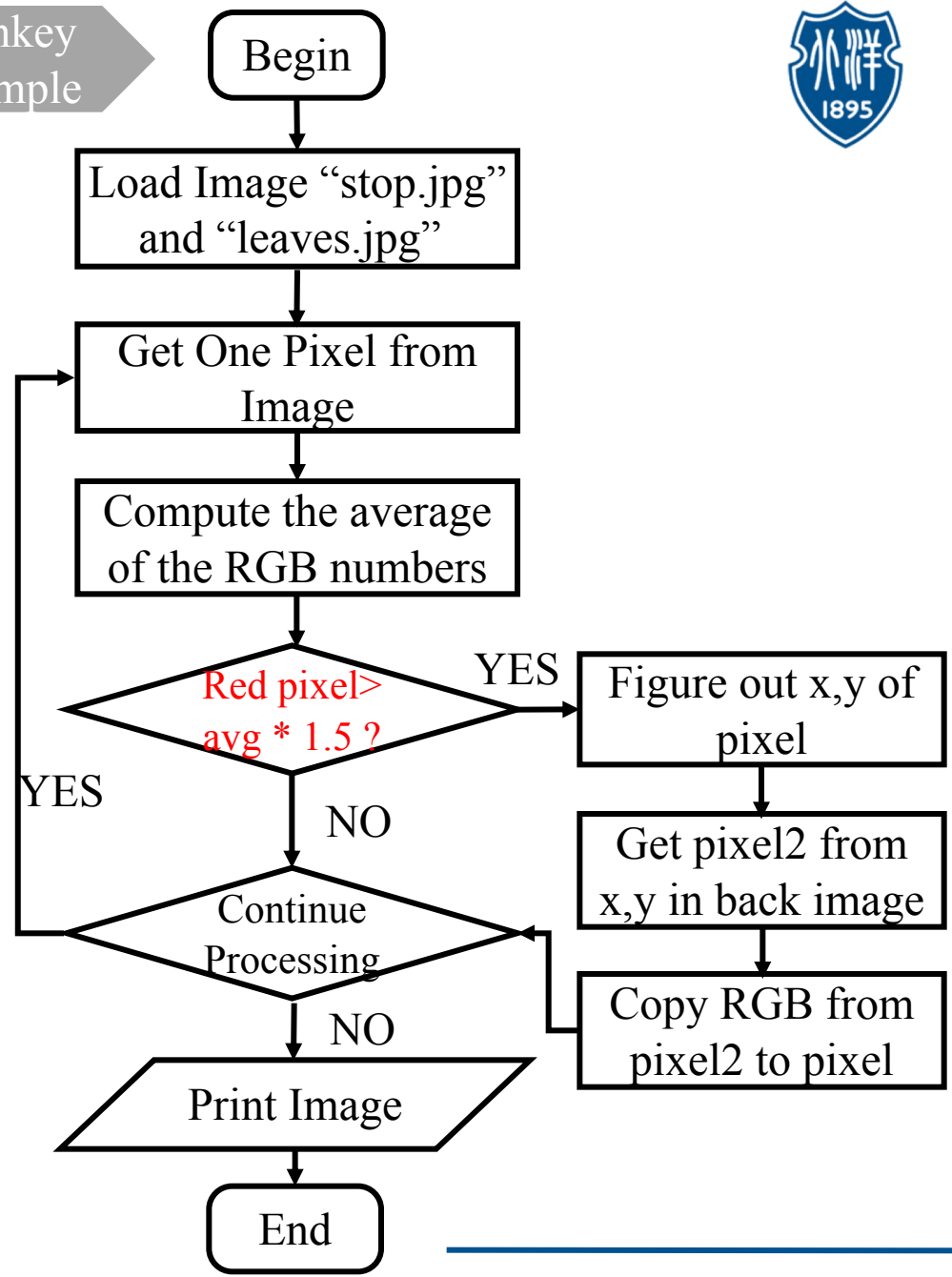


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



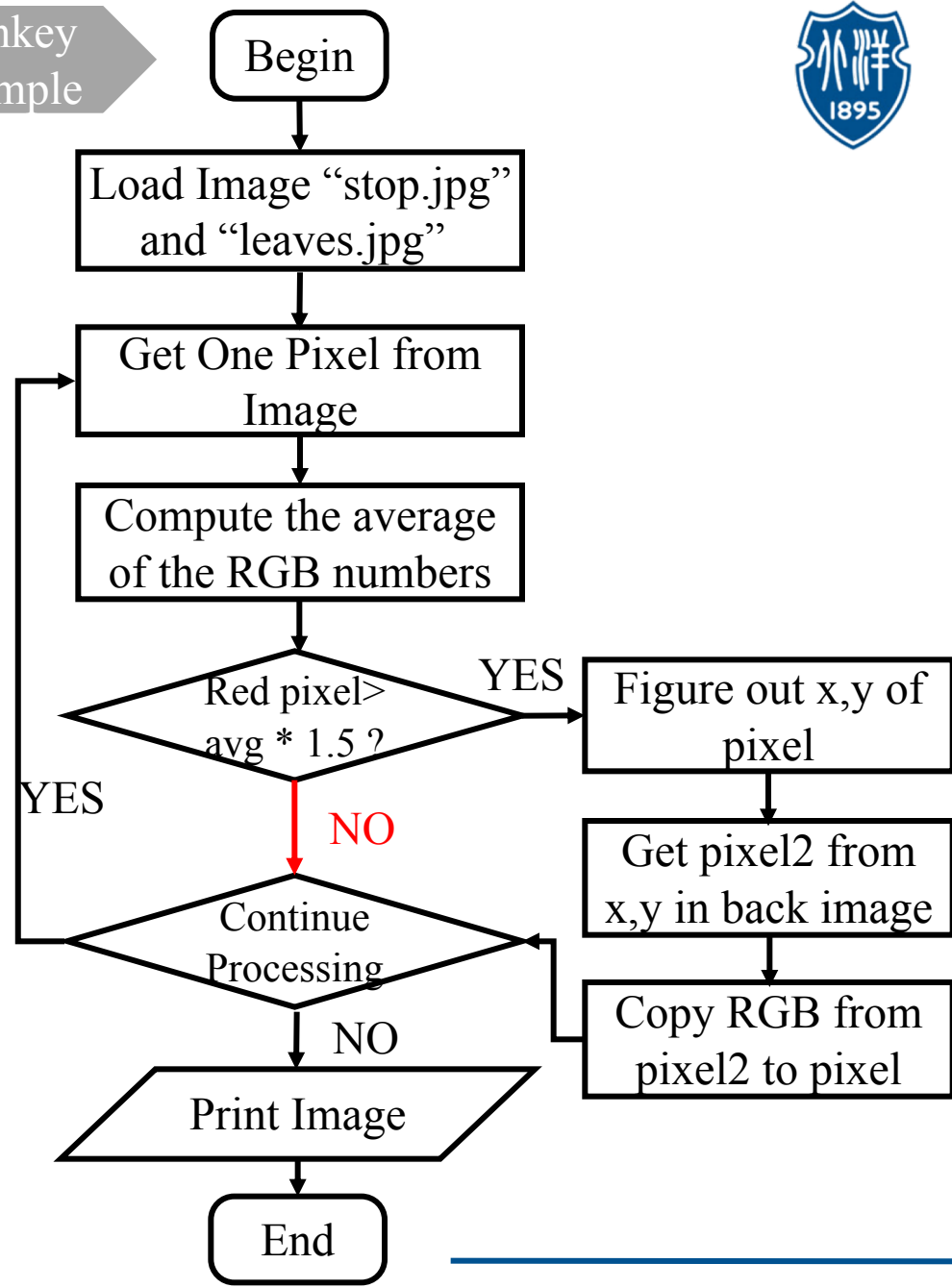


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



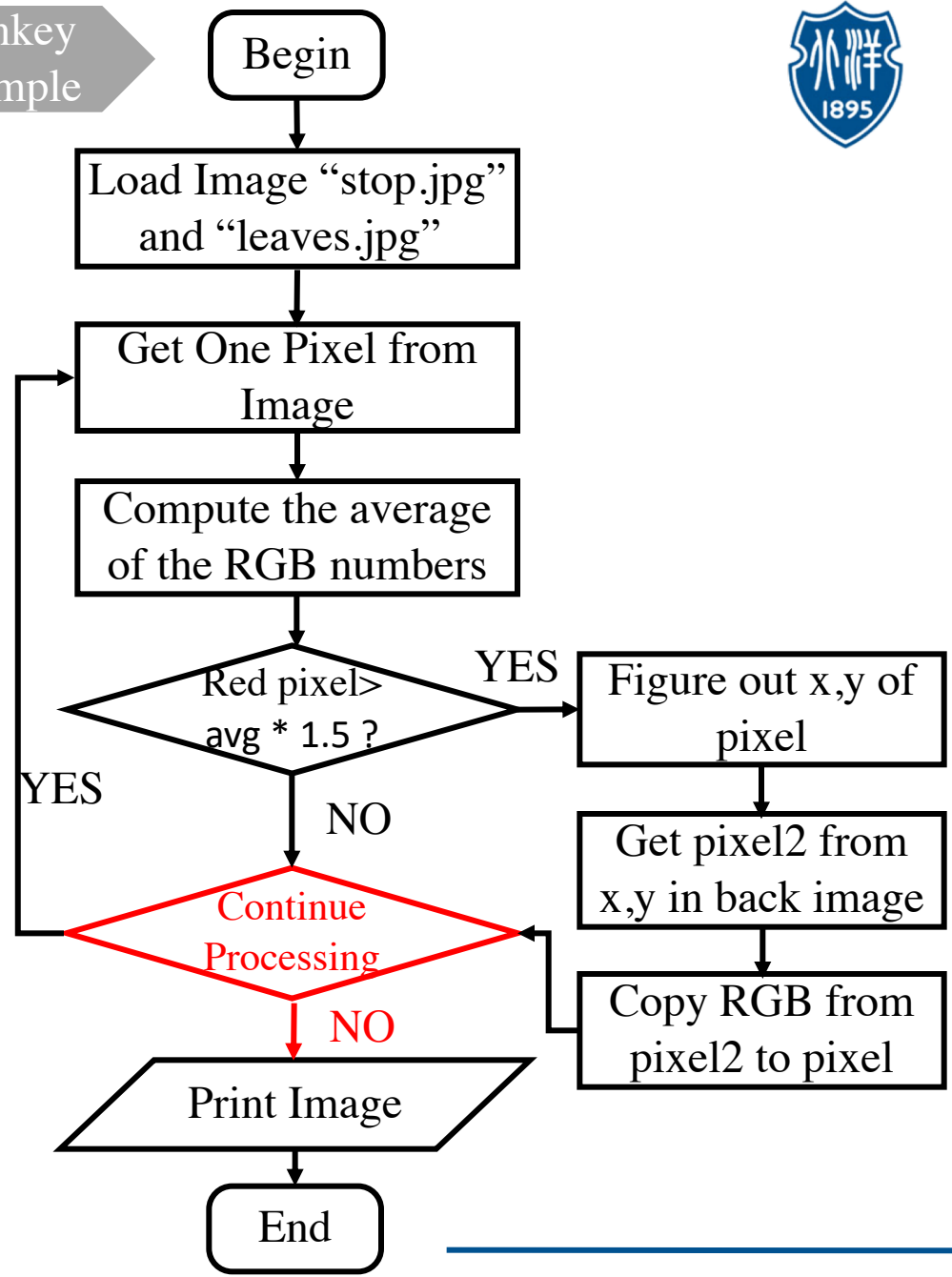


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



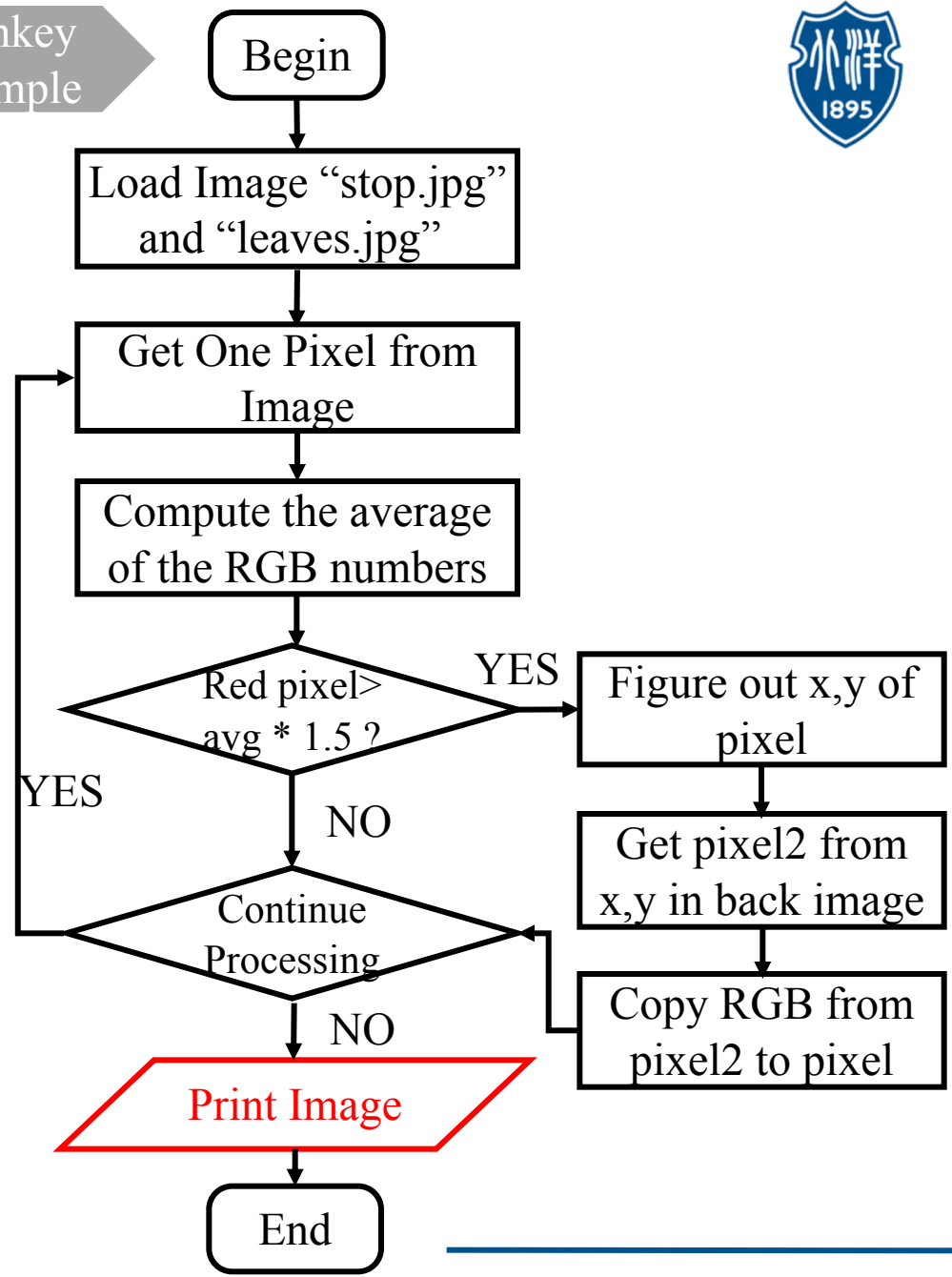


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```



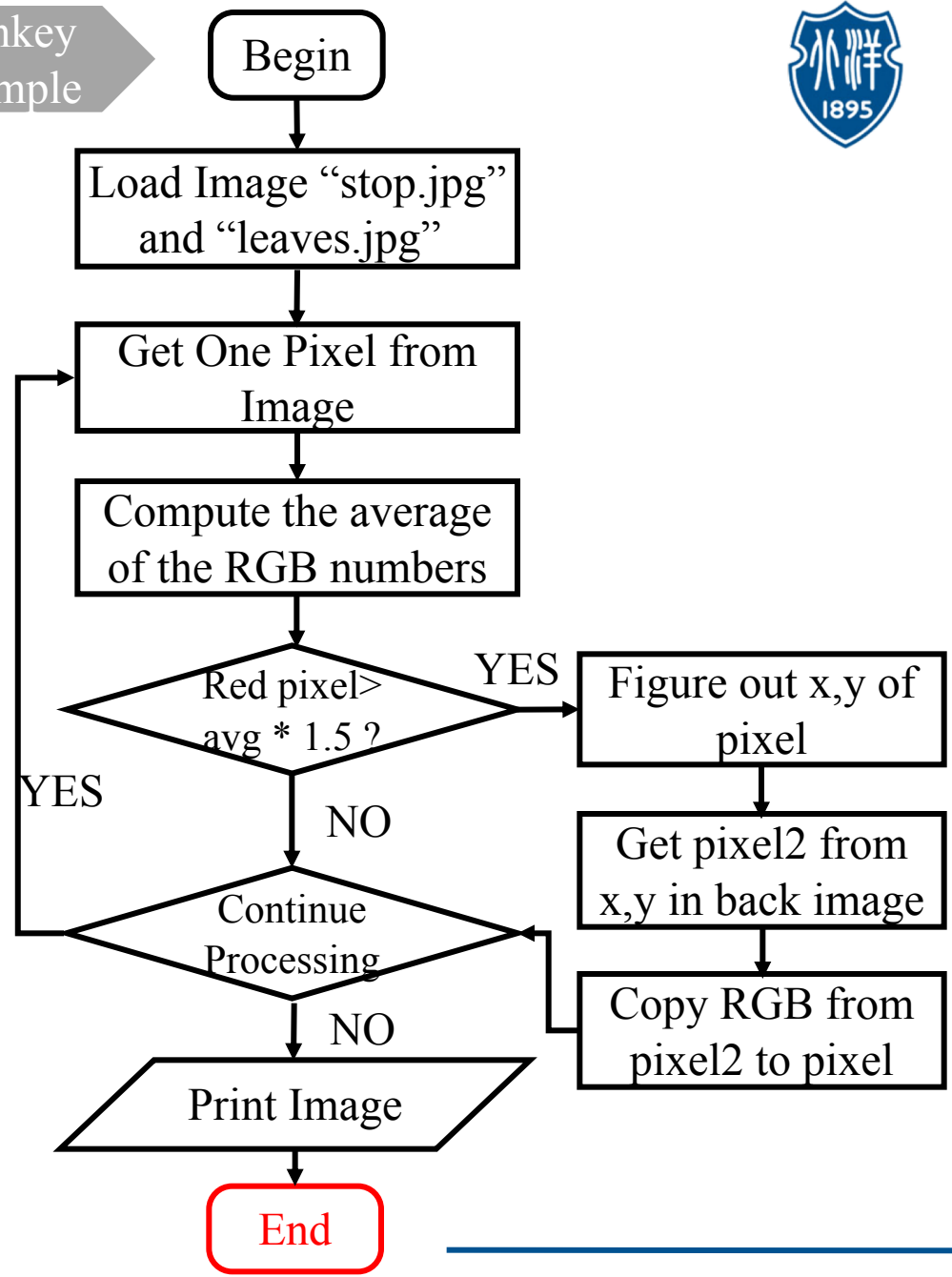


Bluescreen Example Code

```

image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");

for (pixel: image) {
    avg = (pixel.getRed() +
pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.5) {
        x = pixel.getX();
        y = pixel.getY();
        // Translate this line to English
        pixel2 = back.getPixel(x, y);
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
    
```







Bluescreen Summary

- Real technique
- 1. Two images: image, back
- 2. Loop over image, detect color area
- 3. Color area: copy over x,y pixel from back to image

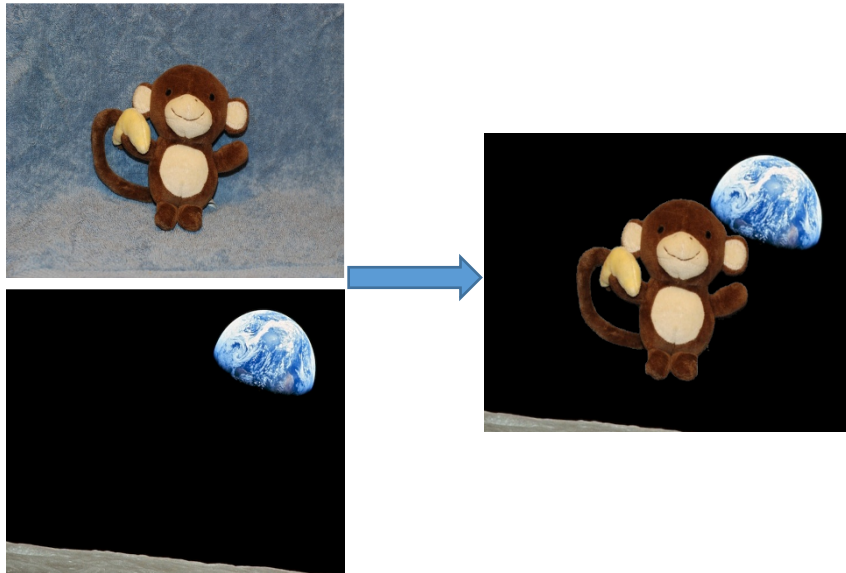


Image



Back



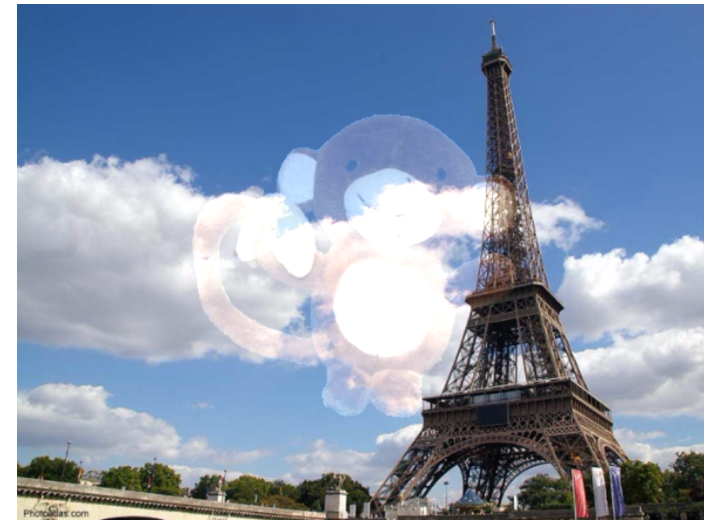


```
image = new SimpleImage("monkey.jpg");
back = new SimpleImage("moon.jpg");

for (pixel: image) {
    avg = (pixel.getRed() + pixel.getGreen()
+ pixel.getBlue())/3;
    if (pixel.getBlue() > avg * 0.92) { //
detect blue-high
        pixel2 = back.getPixel(pixel.getX(),
pixel.getY());
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
```

Image Blend Bluescreen
Multi-Pass Post-Processing
Paint

Image



Back



Image**Back**

There are some problems:

1. The back image is smaller than the main image.
2. What color should we detect ?
3. How to blend the colors from the two images instead of replacing entire pixels.



1. The back image is smaller than the main image.
2. What color should we detect ?
3. How to blend the colors from the two images instead of replacing entire pixels.

```
image = new SimpleImage("monkey.jpg");
back = new SimpleImage("paris.jpg");
back.setAsBig(image);
print(back);
for (pixel: image) {
    avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getBlue() < avg * 0.9) { // detect monkey (blue-low)
        div = 2; // monkey is divided by this variable below
        pixel2 = back.getPixel(pixel.getX(), pixel.getY());
        pixel2.setRed(pixel2.getRed() + pixel.getRed()/div);
        pixel2.setGreen(pixel2.getGreen() + pixel.getGreen()/div);
        pixel2.setBlue(pixel2.getBlue() + pixel.getBlue()/div);
    }
}
print(back);
```




Image(The stop.jpg)



Back(The leaves.jpg)



Back2(The yosemite.jpg)



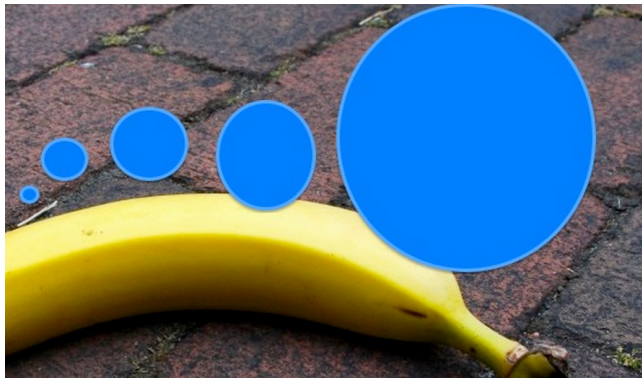
- Make multiple changes to the image: change red and change blue
- Provided code detects red, put in leaves.jpg (back)
- Add a second if-statement after the first (copy/paste)
- Detect blue pixels, put in yosemite.jpg (back2)
- The second if follows the first inside the loop
- They are not inside one another, they are siblings
- They are at the same indent level
- Three pairs `{..}`: 1x for-loop, 2x if-statements



```
image = new SimpleImage("stop.jpg");
back = new SimpleImage("leaves.jpg");
back.setAsBig(image);
back2 = new SimpleImage("yosemite.jpg");
back2.setAsBig(image);
for (pixel: image) {
    avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue())/3;
    if (pixel.getRed() > avg * 1.7) { // select red (back)
        pixel2 = back.getPixel(pixel.getX(), pixel.getY());
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
    if (pixel.getBlue() > avg * 1.2) { // select blue (back2)
        pixel2 = back2.getPixel(pixel.getX(), pixel.getY());
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
```



Suppose we want to show the banana having a dream about paris. But we don't want to replace all the red brick. This image is not well suited to bluescreen (banana vs. brick, too much red). So, what should we do?



- We can "fix it in post" (Hollywood expression)
- I created banana2.jpg as a copy to edit
- Put in blue circles artistically (demo)
- Now the code can just look for blue
- Can also use this to "fix" little areas discretely in the original image that are giving your bluescreen selection logic problems (like when you have little speckles you can't get rid of)



```
image = new SimpleImage("banana2.jpg");
back = new SimpleImage("paris.jpg");
back.setAsBig(image);
for (pixel: image) {
    avg = (pixel.getRed() + pixel.getGreen() +
pixel.getBlue())/3;

    if (pixel.getBlue() > avg * 1.3) { // select blue
        pixel2 = back.getPixel(pixel.getX(), pixel.getY());
        pixel.setRed(pixel2.getRed());
        pixel.setGreen(pixel2.getGreen());
        pixel.setBlue(pixel2.getBlue());
    }
}
print(image);
```





Thank You!

Q&A