计算原理导论

Introduction to Computing Principles

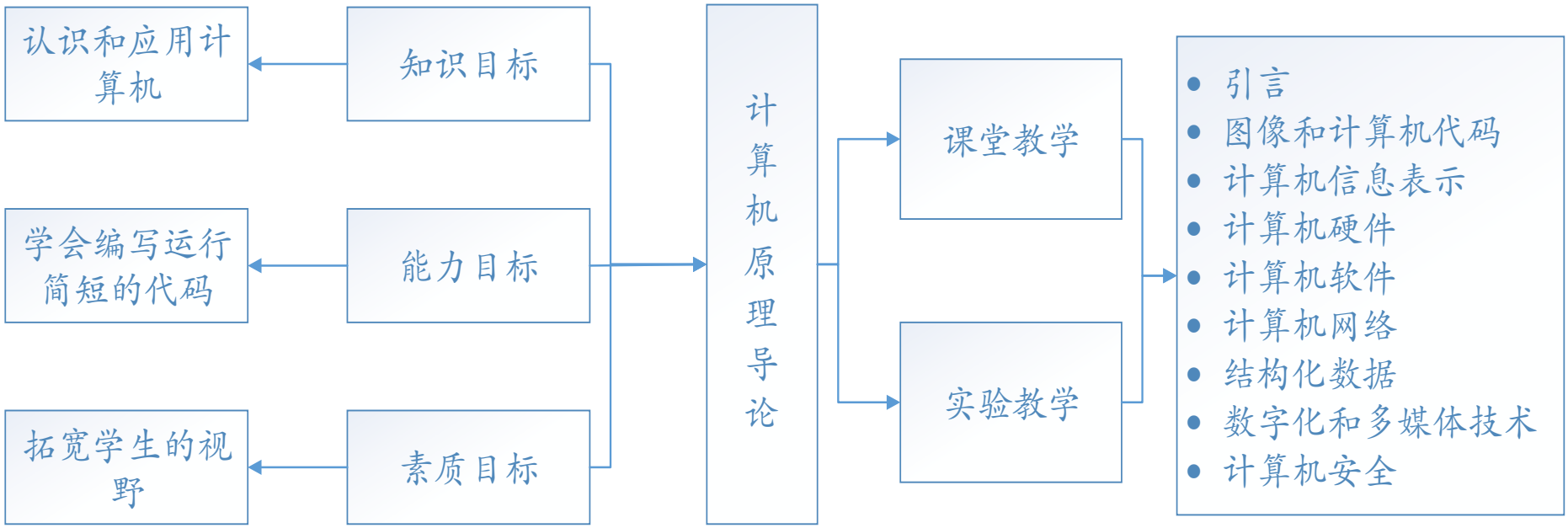
刘志磊
天津大学　计算机科学与技术学院

课程信息

2016–2017 学年第一学期日程安排表

日期	08.22-08.28	08.29-09.04	09.05-09.11	09.12-09.18	09.19-08.25	09.26-10.02	10.03-10.09	10.10-10.16	10.17-10.23	10.24-10.30	10.31-11.06	11.07-11.13	11.14-11.20	11.21-11.27	11.28-12.04	12.05-12.11	12.12-12.18	12.19-12.25	12.26-01.01
周数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
周一			上课	上课	上机	上机		上课	上机	上机	上课	上课	上机	上机	上课	上课	上机	上机	
周二																			
周三			上课		上机		上课		上机		上课		上机		上课		上机		
周四																			
周五																			
周六																			
周日							补课												
备注																			

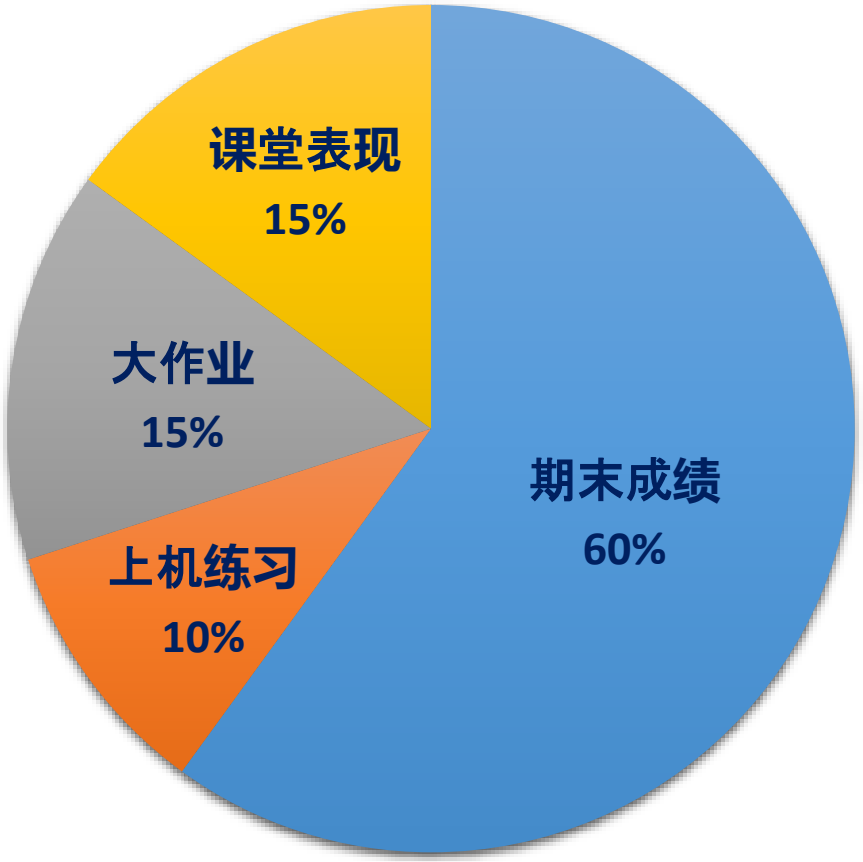
课程框架与章节


认识和应用计算机
学会编写运行简短的代码
拓宽学生的视野
知识目标
能力目标
素质目标
计算机原理导论
课堂教学
实验教学
引言
图像和计算机代码
计算机信息表示
计算机硬件
计算机软件
计算机网络
结构化数据
数字化和多媒体技术
计算机安全

参考资料

- Stanford Online Lagunita
  - https://lagunita.stanford.edu/courses/Engineering/CS101/Summer2014/
- The Stanford Website by Nick
  - http://web.stanford.edu/class/cs101/
- MOOC 慕课平台
  - http://mooc.guokr.com/course/406/Computer-Science-101/
- Fundamentals of Computer Science (3rd edition): By Behrouz Forouzan, 2014

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+2016*******+张三+ImageEx1

2. 三次翻转课堂

3. 三次大作业

4. 三次缺勤取消考试资格

Beginning

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

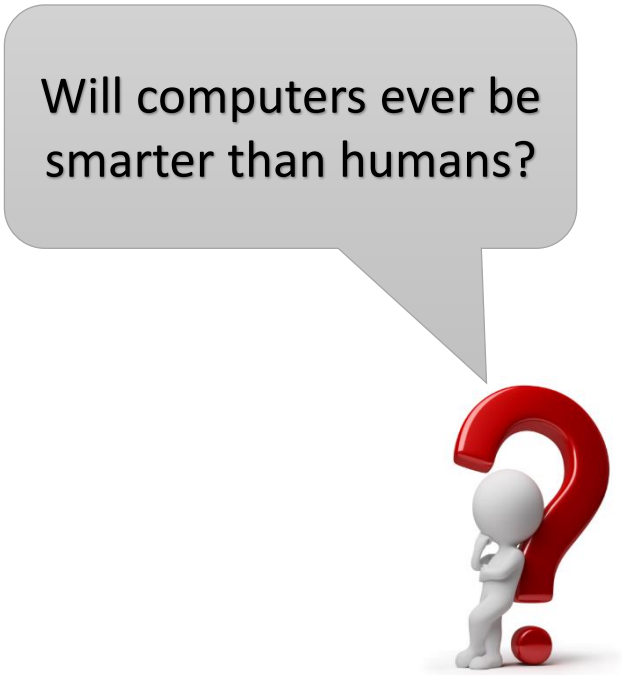
Computer = Powerful + Stupid

Powerful

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

Stupid

-- Operations are simple and mechanical
-- Lacks " insight "



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

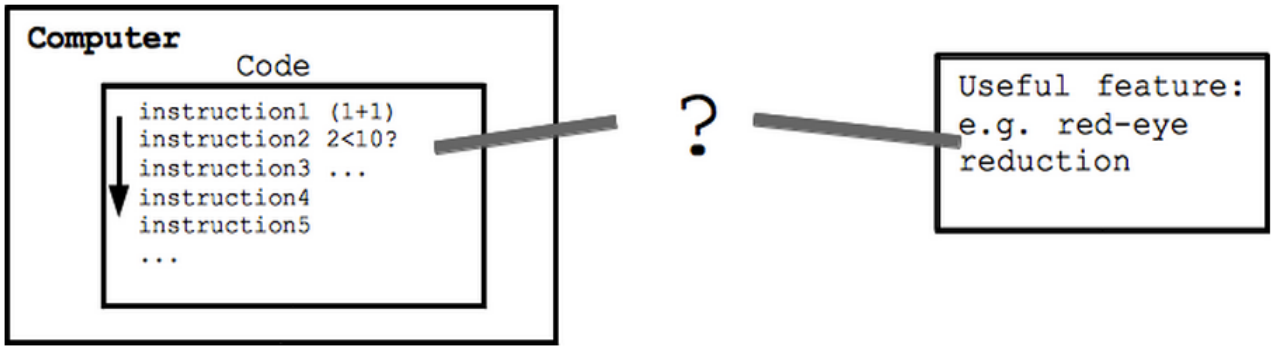


Image and code

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

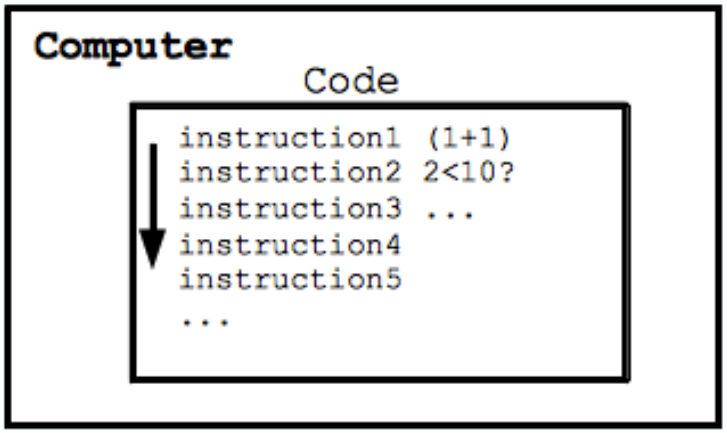


Image and code

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

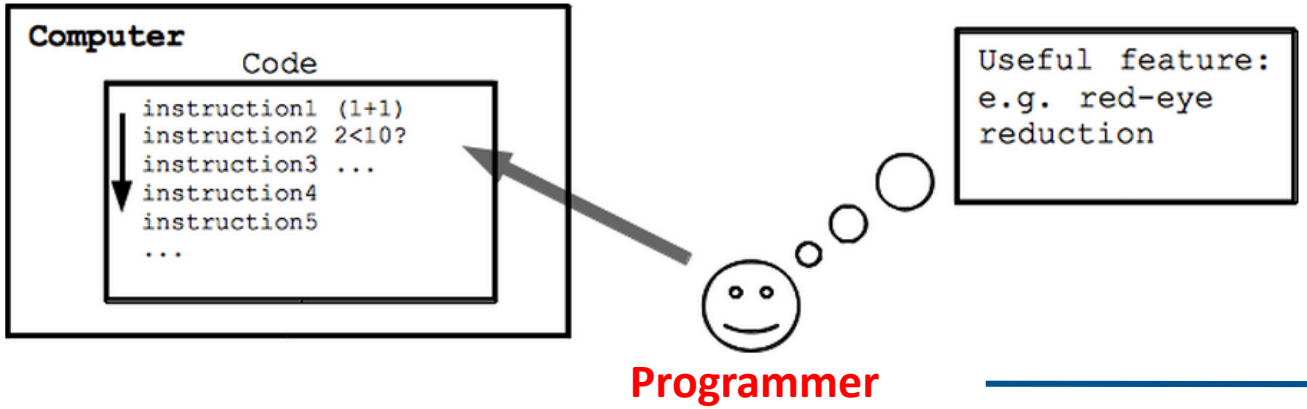


Image and code

- 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 ≠ 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
```

Run

code1-1

- **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**

Excise

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

```
a
Error:prlnt is not defined
```

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

```
Error:Unexpected token ILLEGAL
```

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

```
Error:missing ) after argument list
```

Excise

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

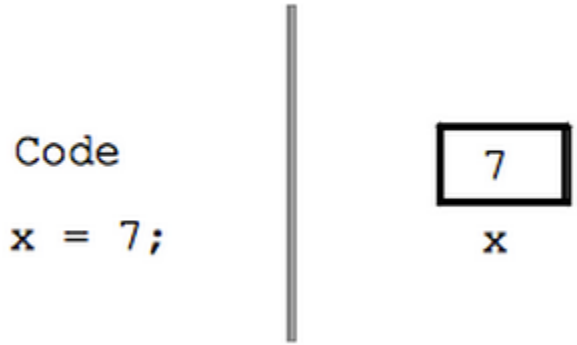


Image and code

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;

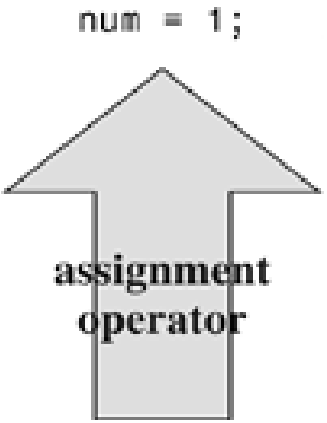


Image and code

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	Invalid Names
`wiggles`	`$Z]**`
`cat2`	`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

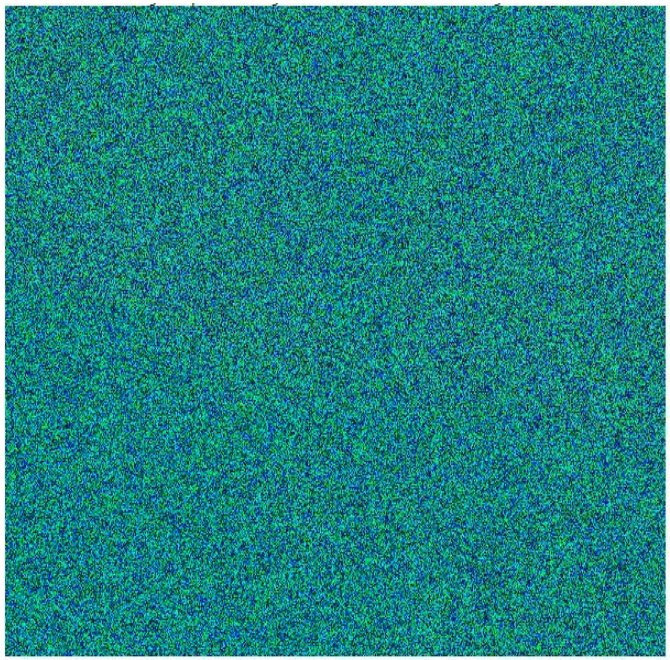
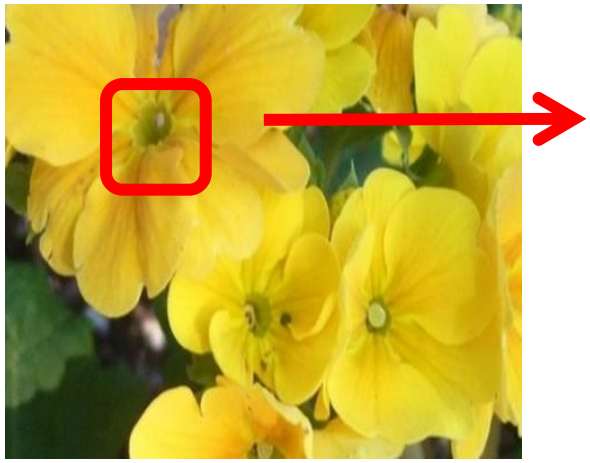


Image Representation in Computer

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

Pixel 像素

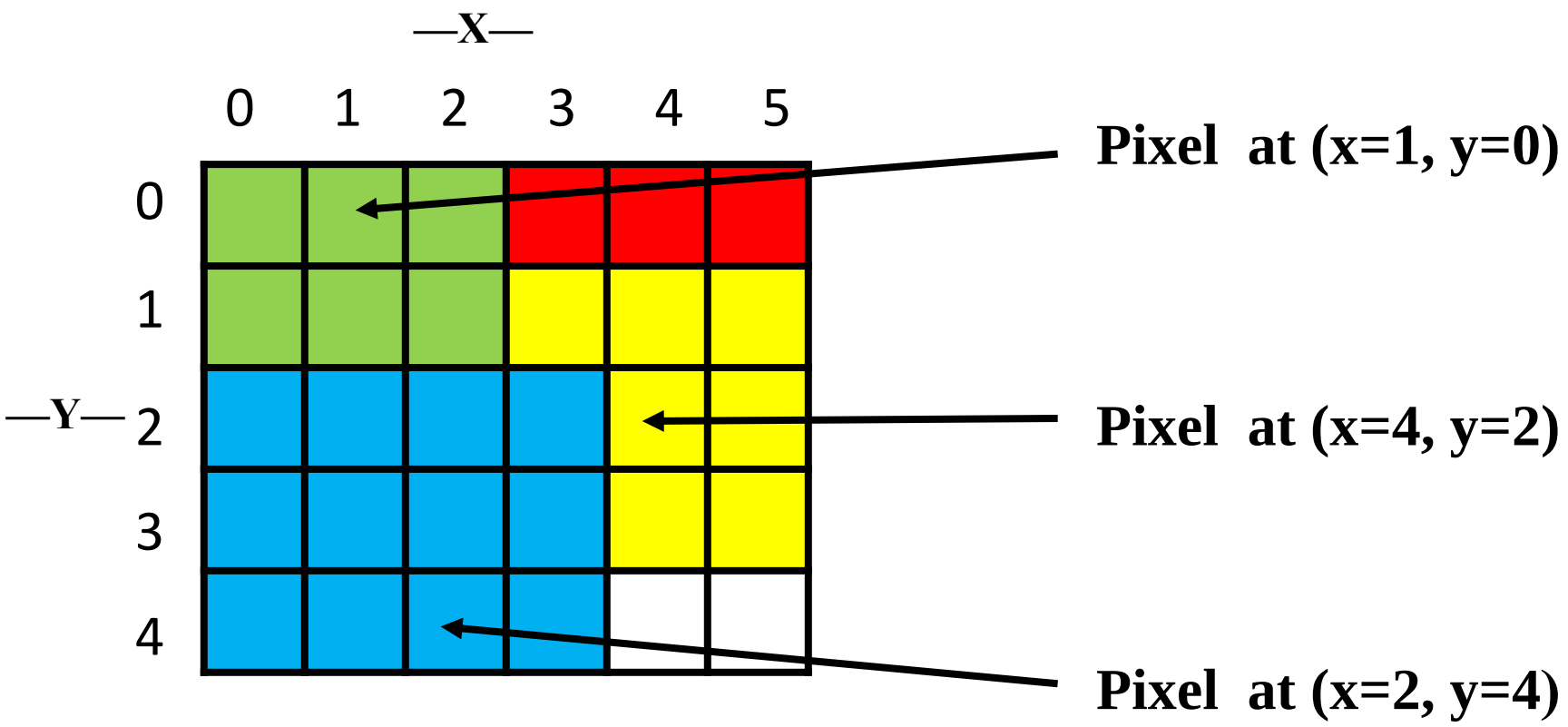


Definition

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 Diagram 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

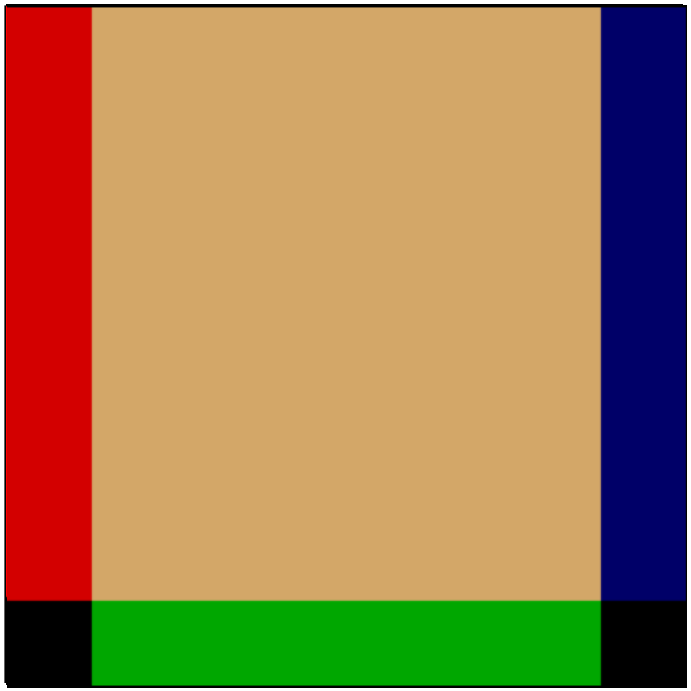
RGB Explorer

Red:

Green:

Blue:

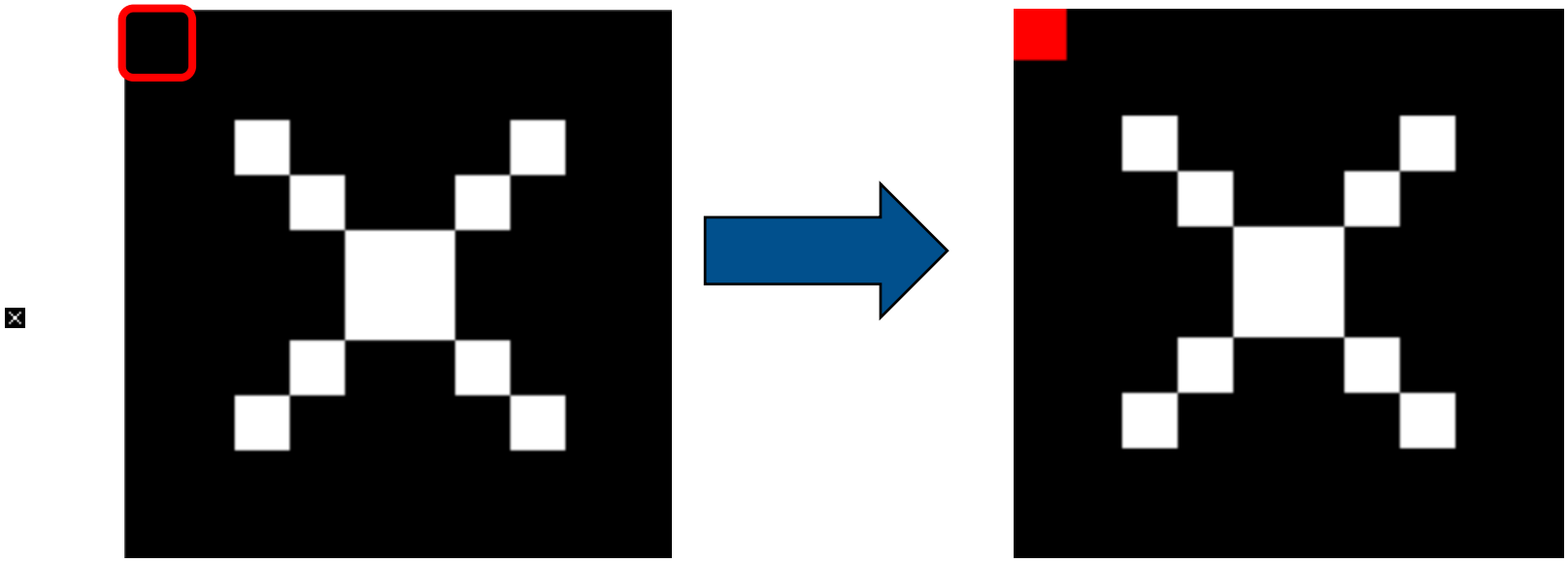
Red: 251 Green: 165 Blue: 154



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

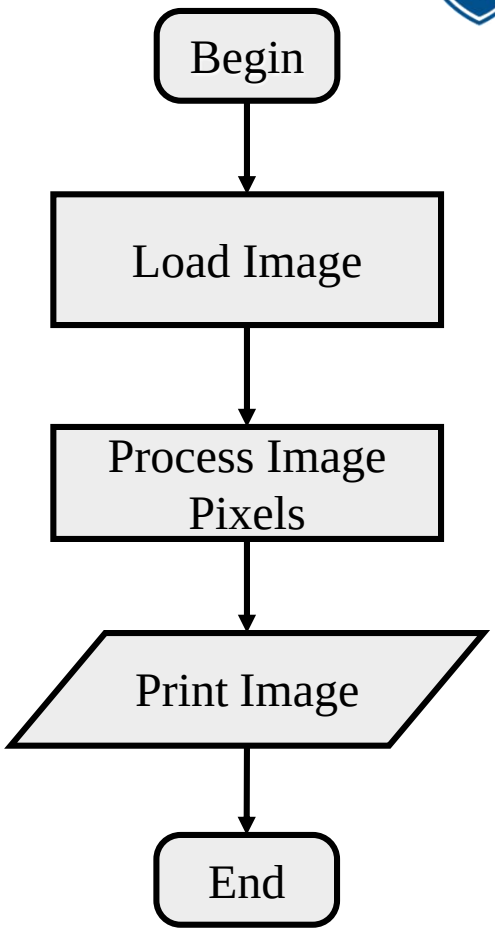
Definition

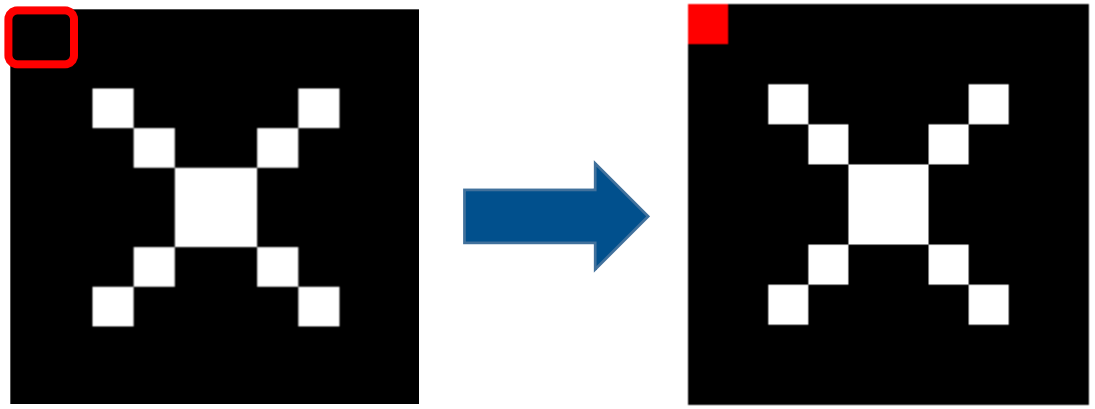
- In computer programming, a function is a sequence of program instructions that perform a specific task, packaged as a unit. In different programming languages, a function may be called a procedure, a subroutine, a method, or a subprogram.

Image Zoom Exercise

getPixel() & setRed()

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

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

print(image);
```





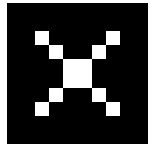
[pixel.setRed(255) Example]

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

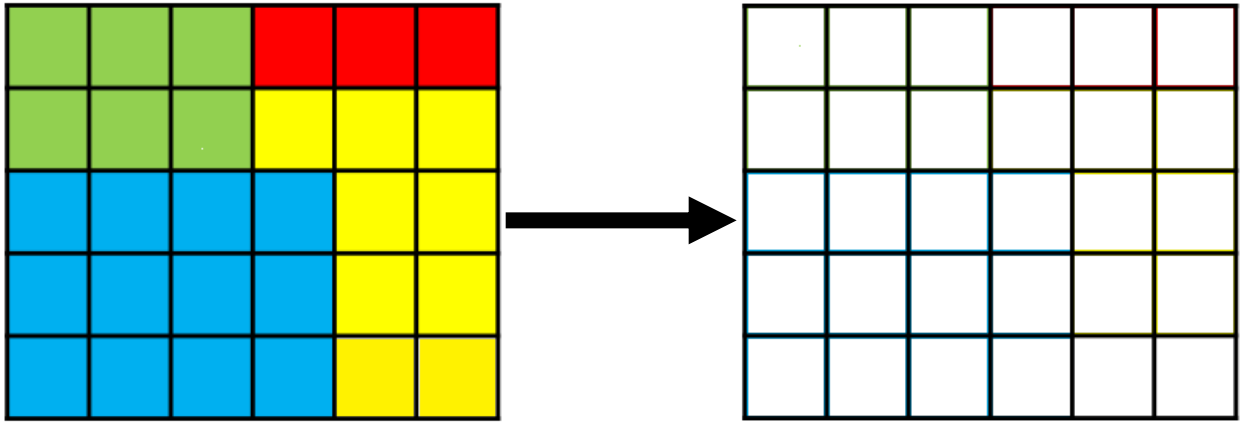
- A Link
  - http://web.stanford.edu/class/cs101/image-functions-reference.html

- Experiments on Pixel (0,0)

Definition

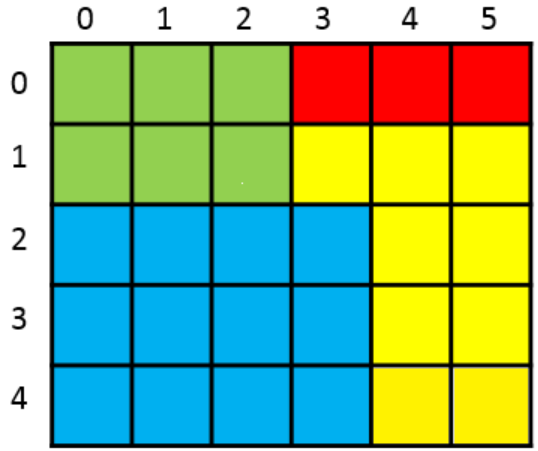
- A loop is a sequence of statements which is specified once but which may be carried out several times in succession. The code "inside" the loop is obeyed a specified number of times, or once for each of a collection of items, or until some condition is met, or indefinitely.

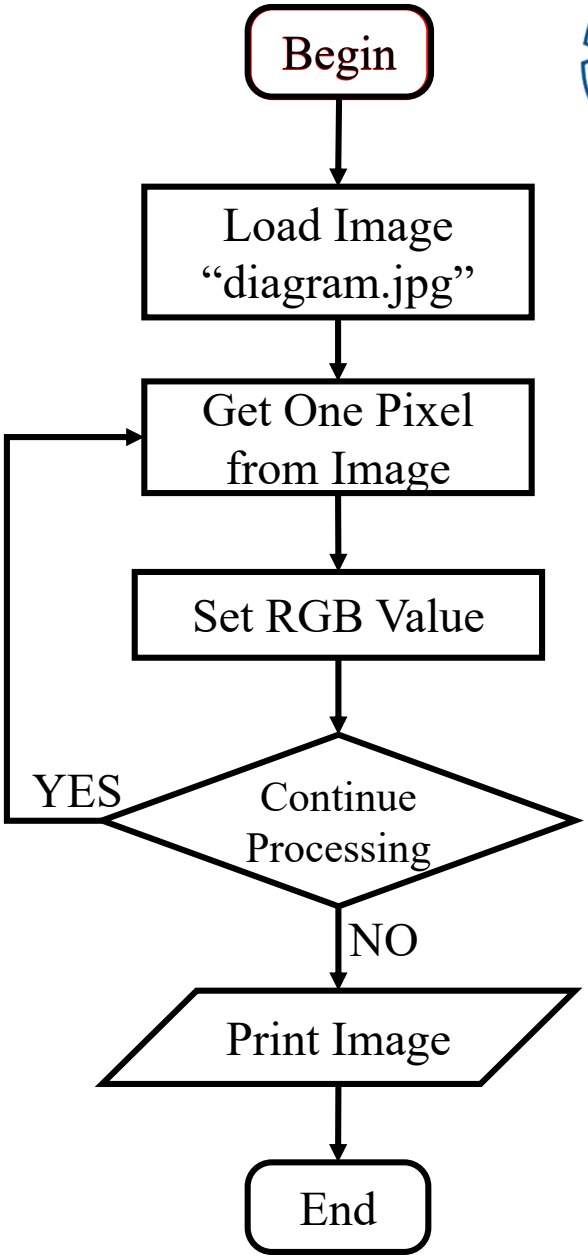
How can we change the color of all the pixels in image?



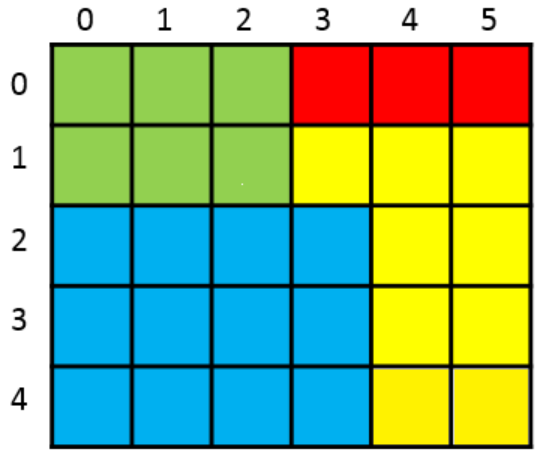
Loops (循环) do some steps "for all these ite

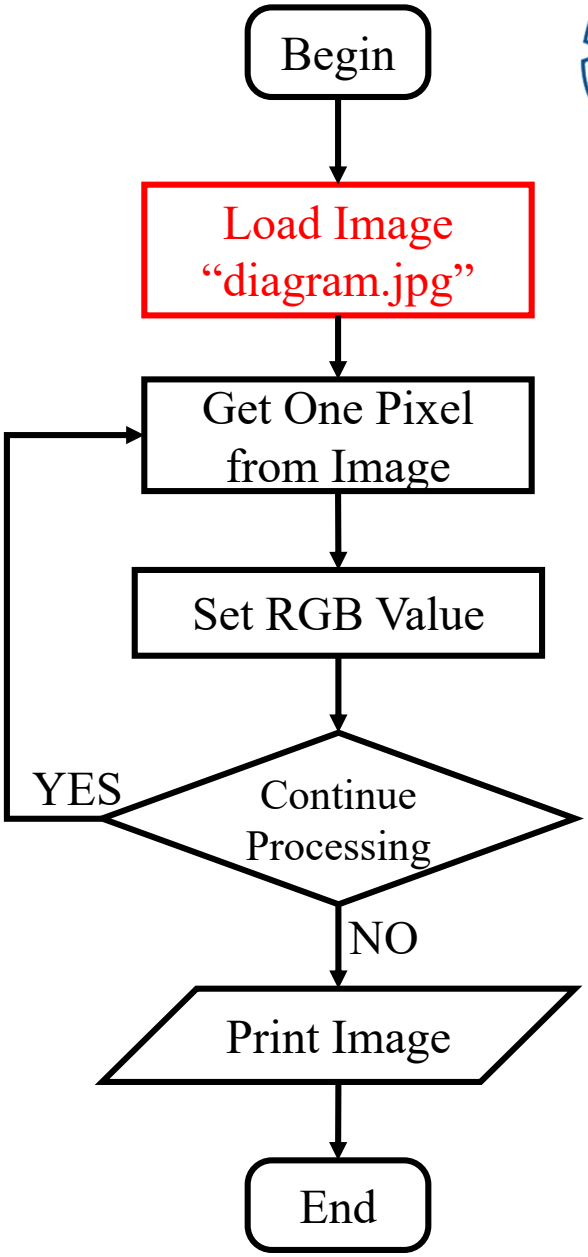
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



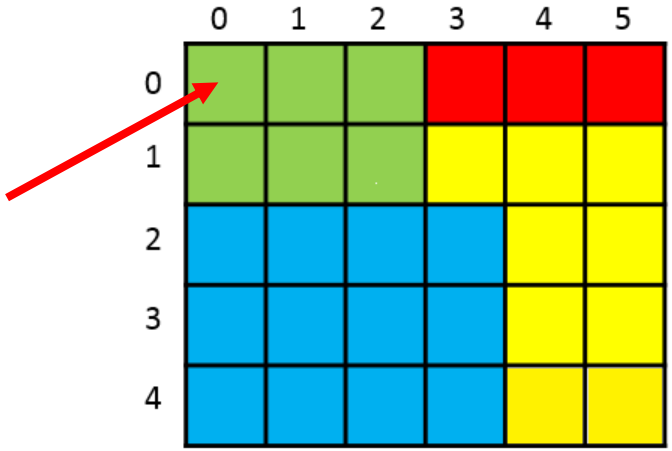


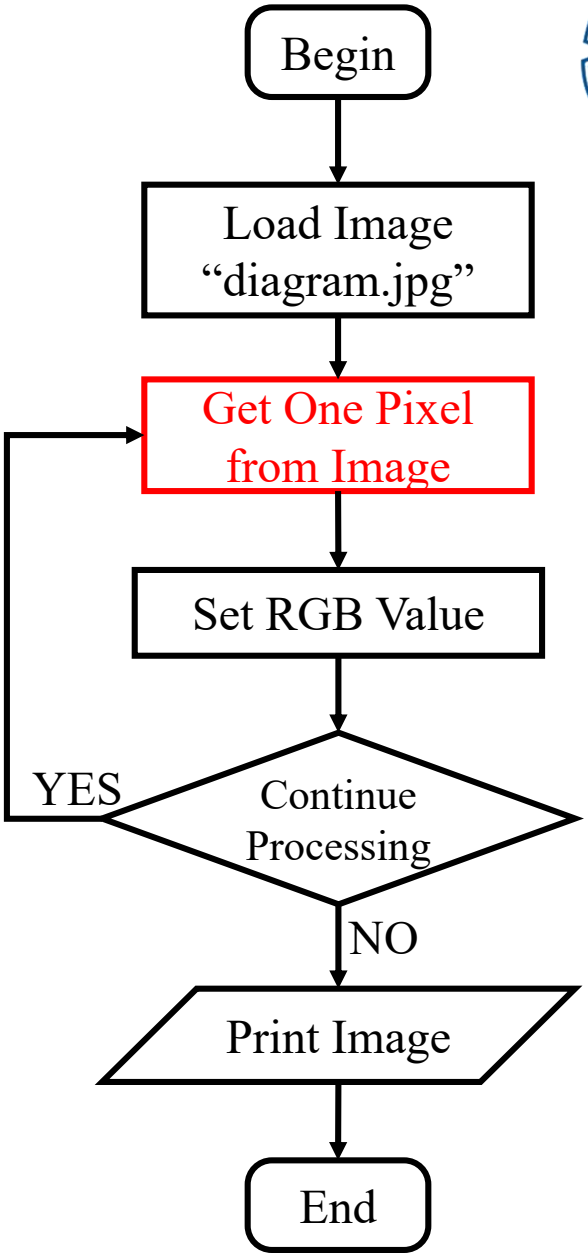
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



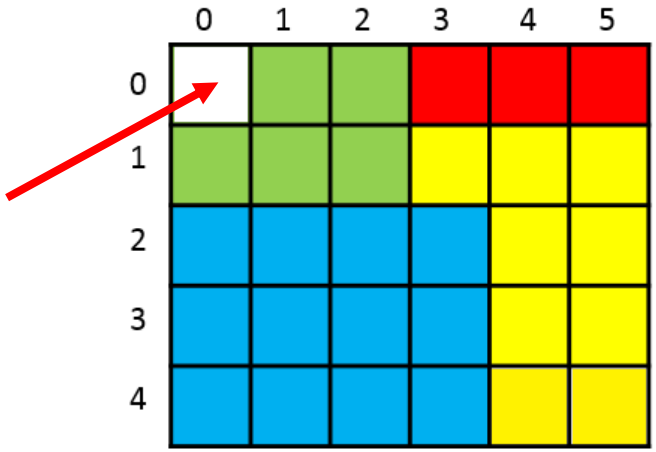


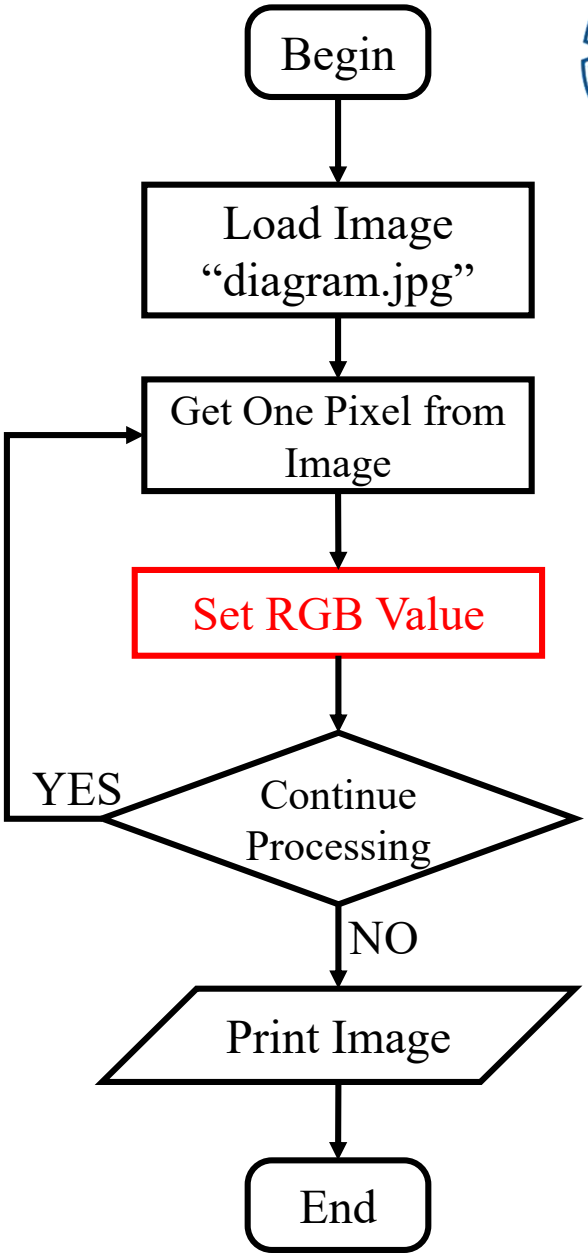
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



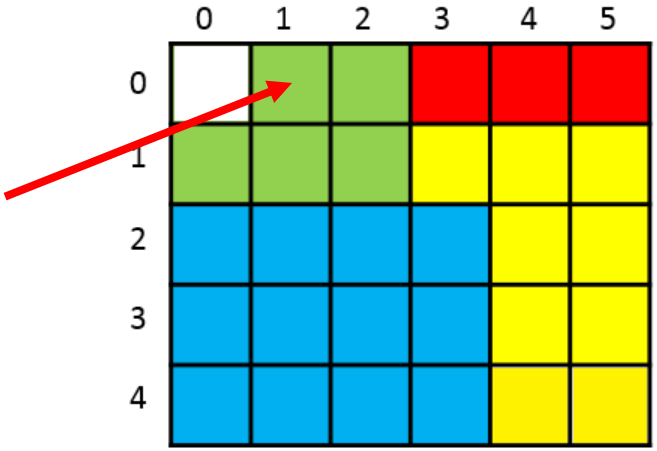


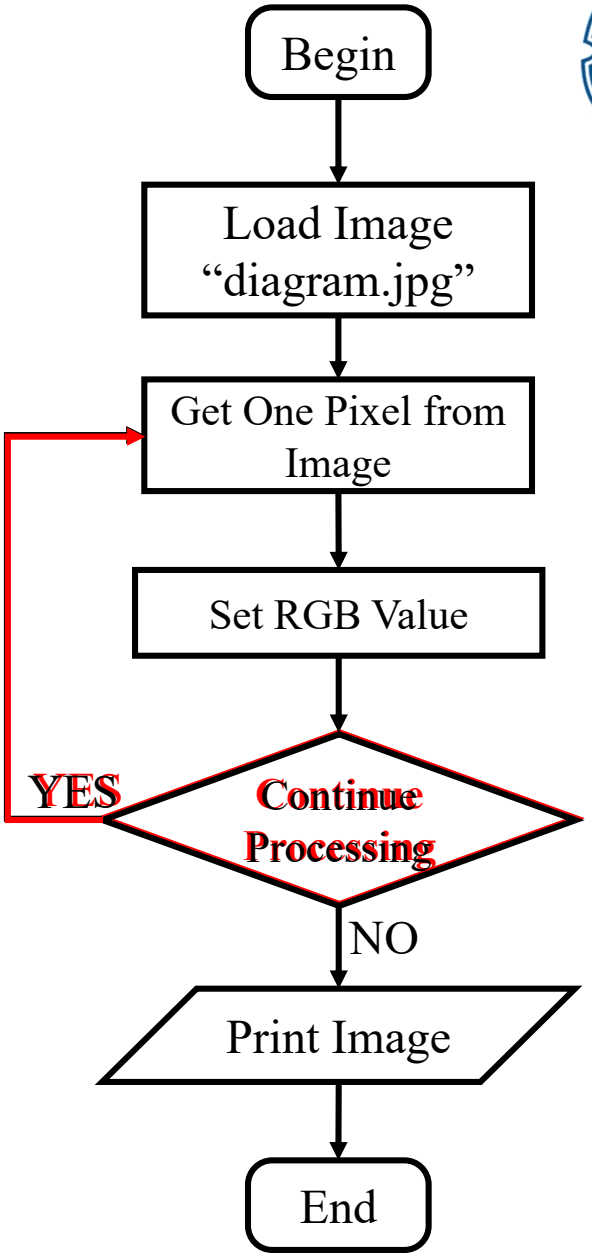
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



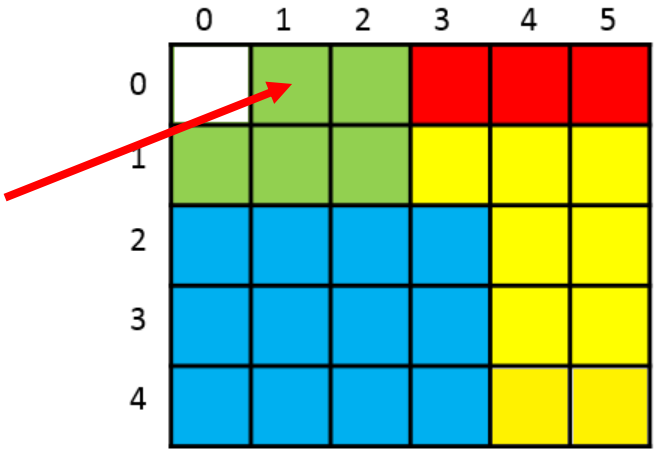


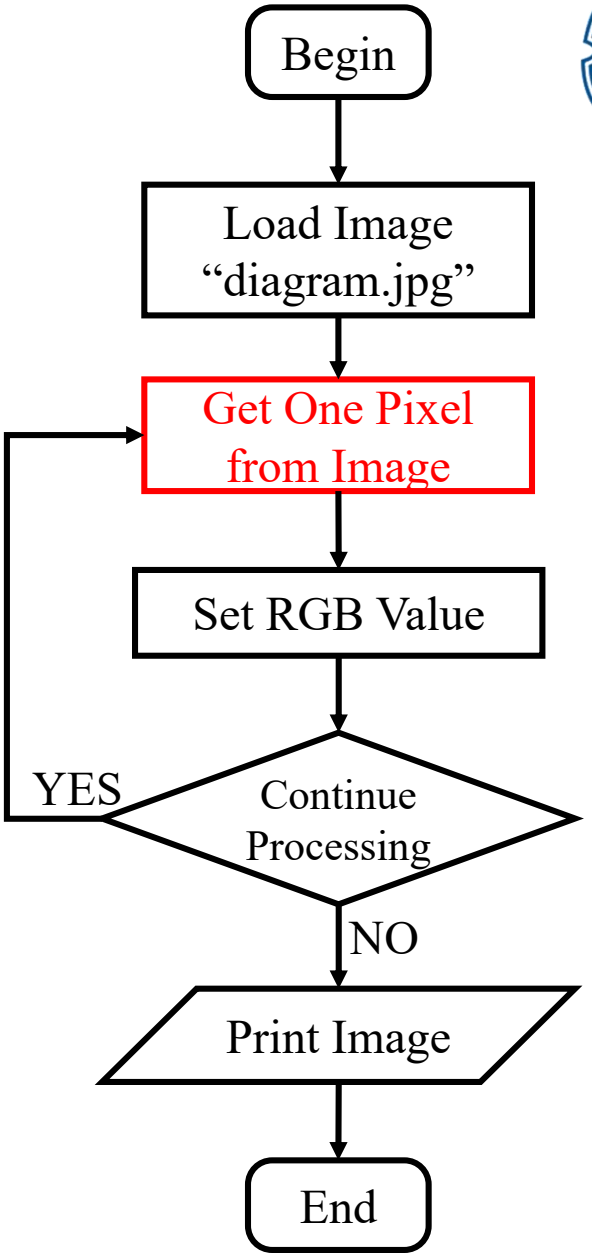
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



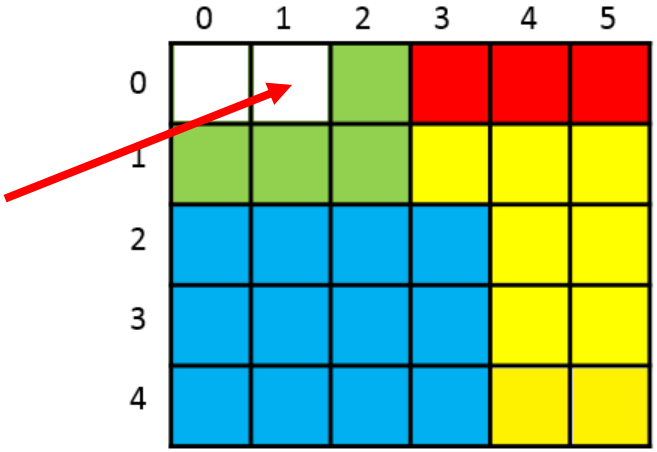


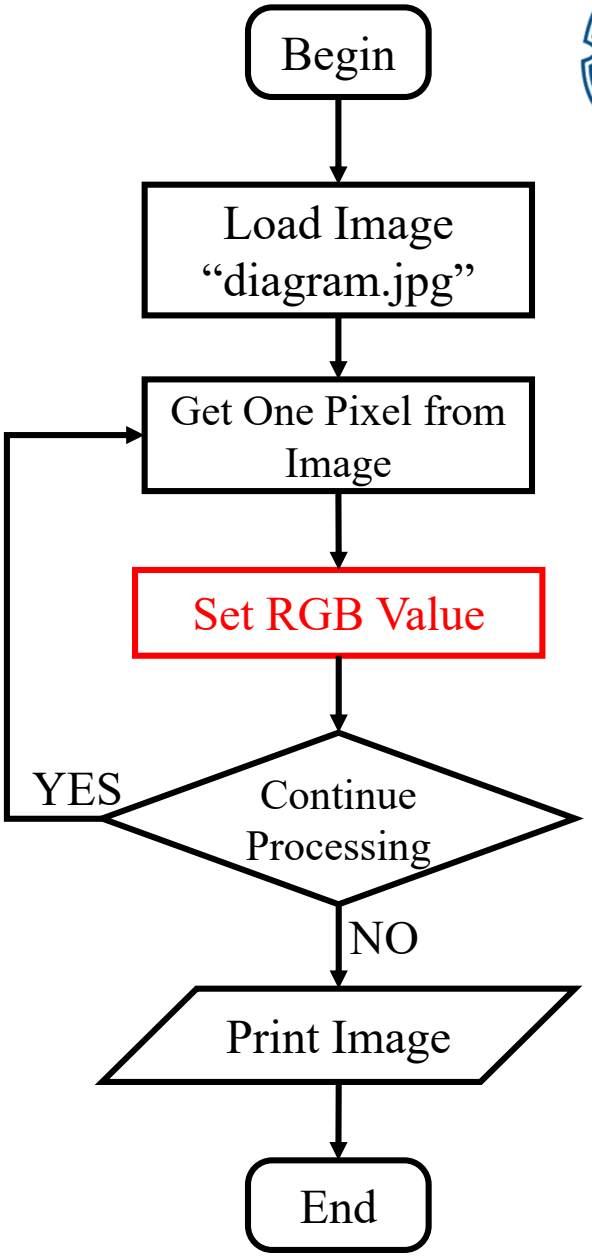
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



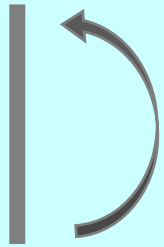


```
image = new SimpleImage("diagram.jpg");

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

print (image);
```

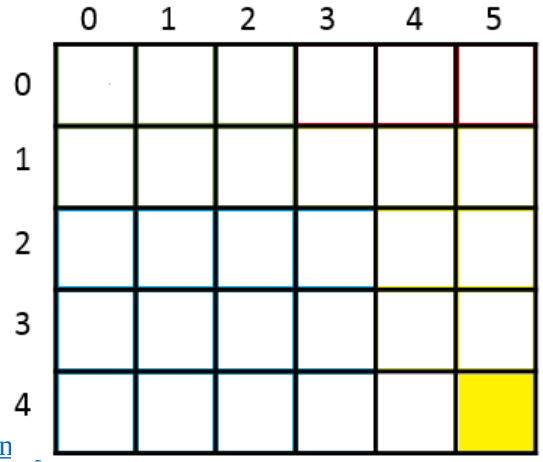


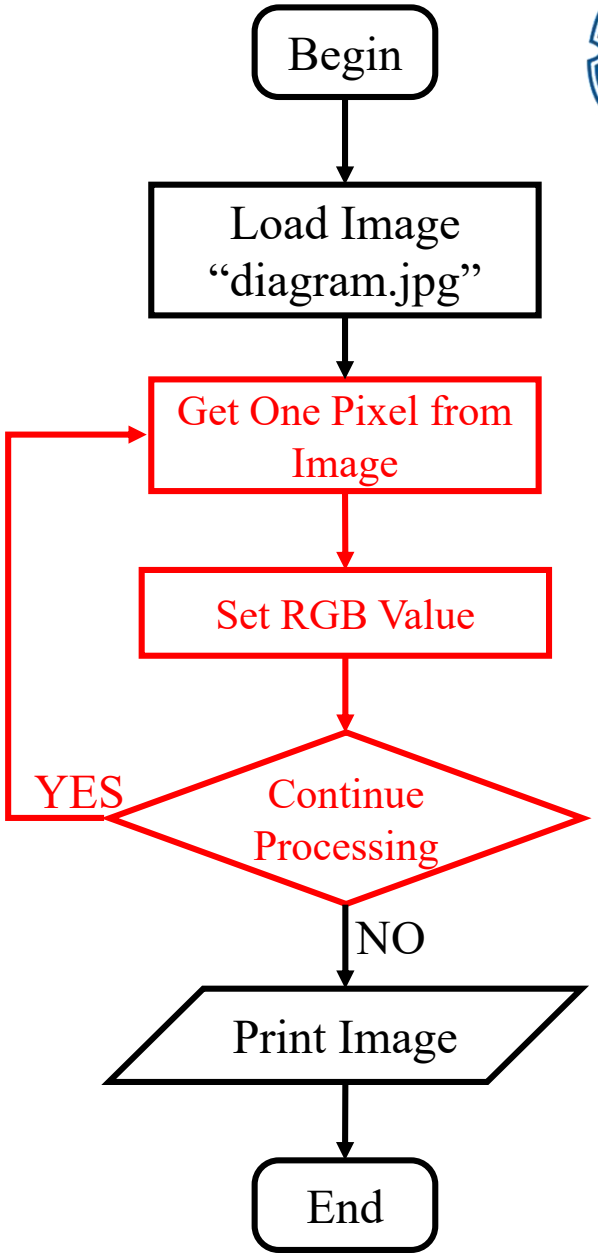


```
image = new SimpleImage("diagram.jpg");

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

print (image);
```

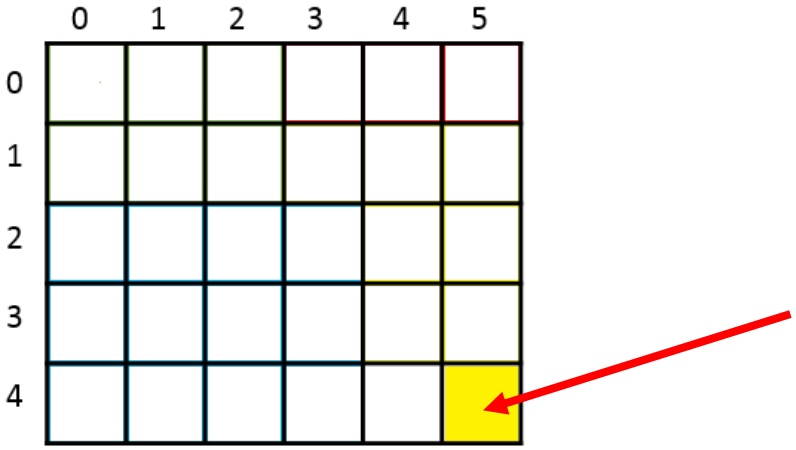


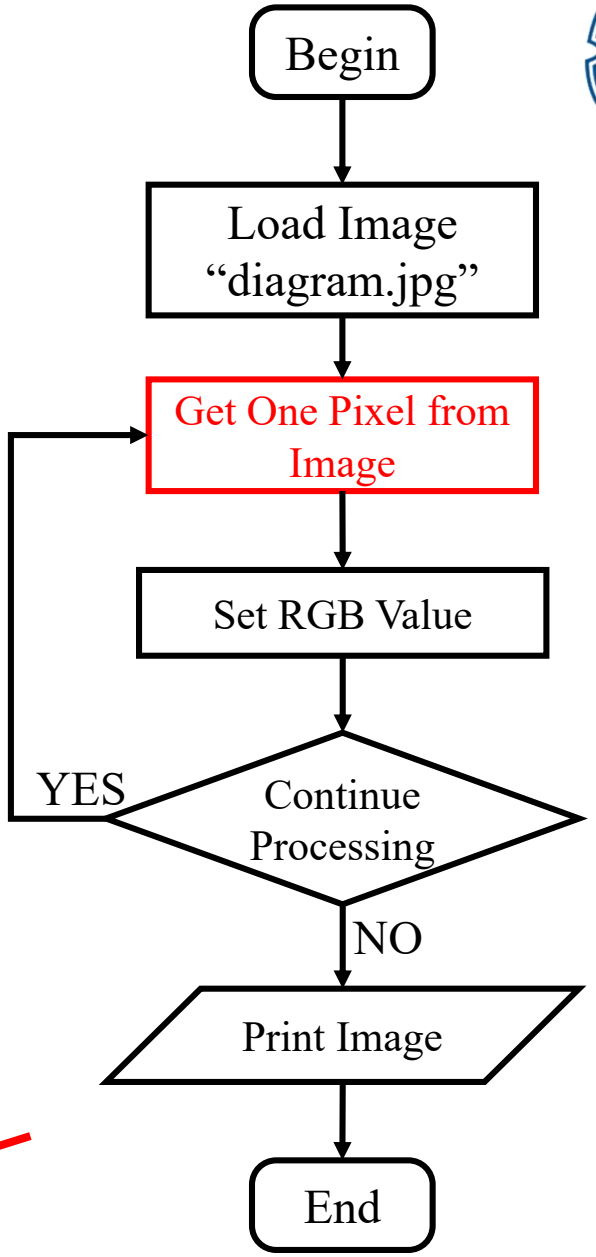
computer runs it again and again, once for each pixel … …



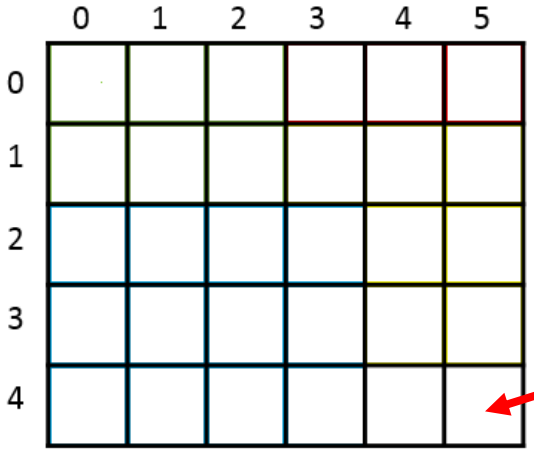


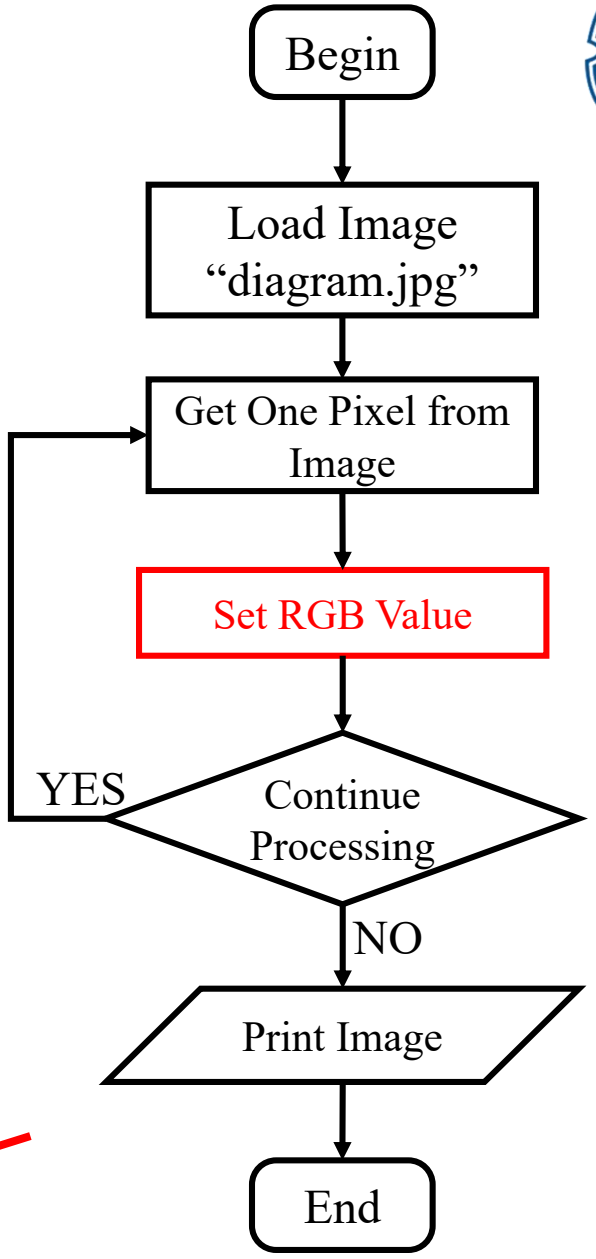
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



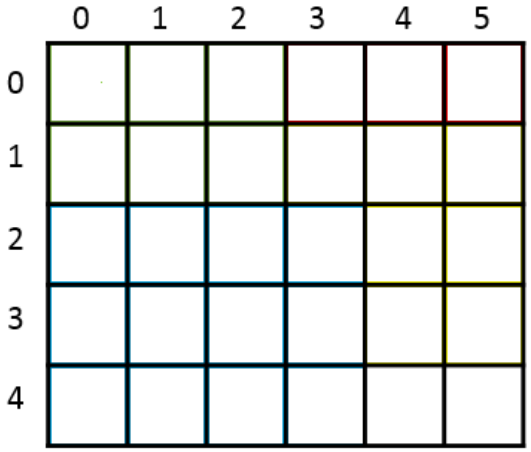


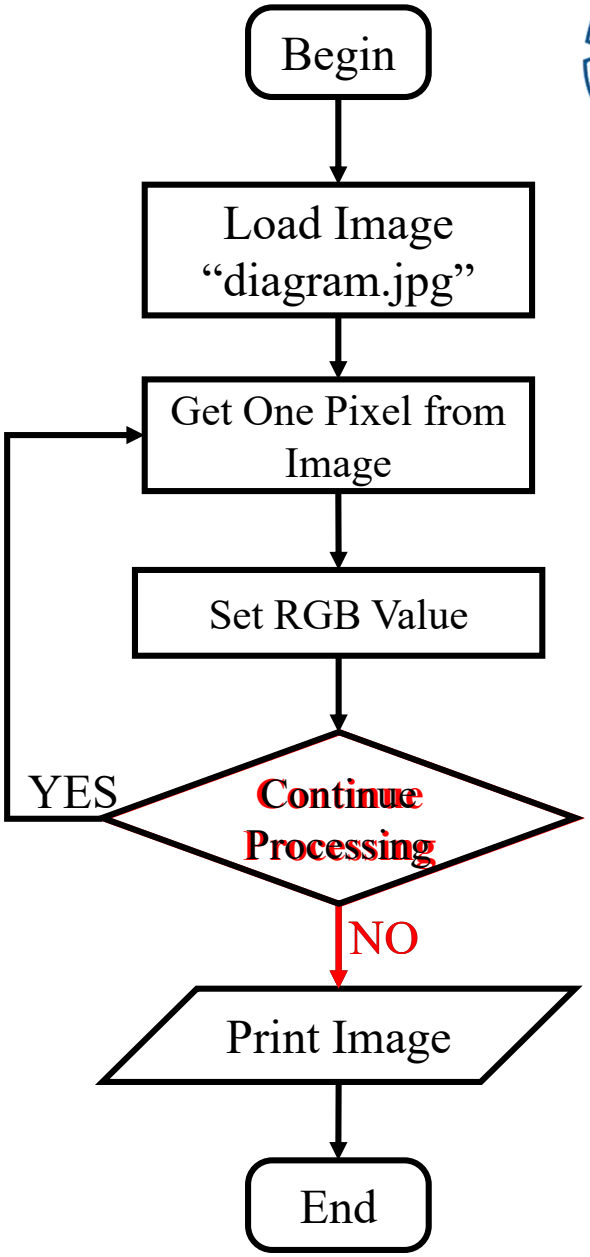
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```



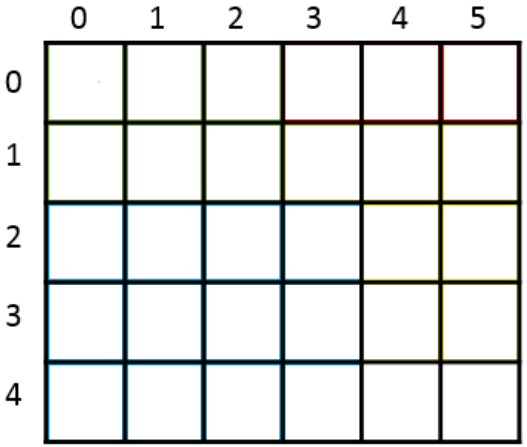


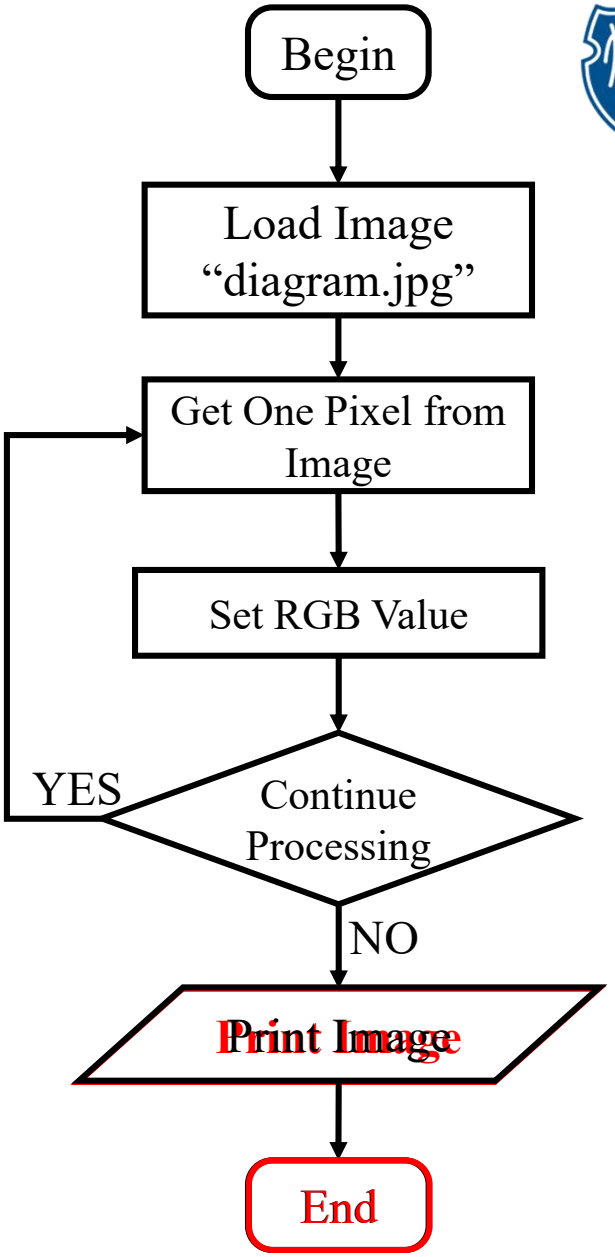
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```





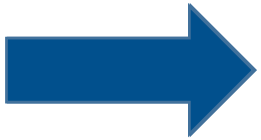
```
image = new SimpleImage("diagram.jpg");

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

print (image);
```

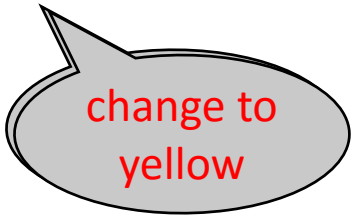




For-Loop Example

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

for (pixel: image) {
  pixel.setRed(255);
  pixel.setGreen(255);
  pixel.setBlue(0);
}

print(image);
```





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:

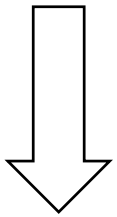
```
print(11 + 31);
```

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 * 5, * 10, * 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 x 2, 10 first x 2, 20 first x 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 both 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.

How to make gray color?

- 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 covert 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 is brightest?

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

	red	green	blue	average
				average = (red + green + 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);
```

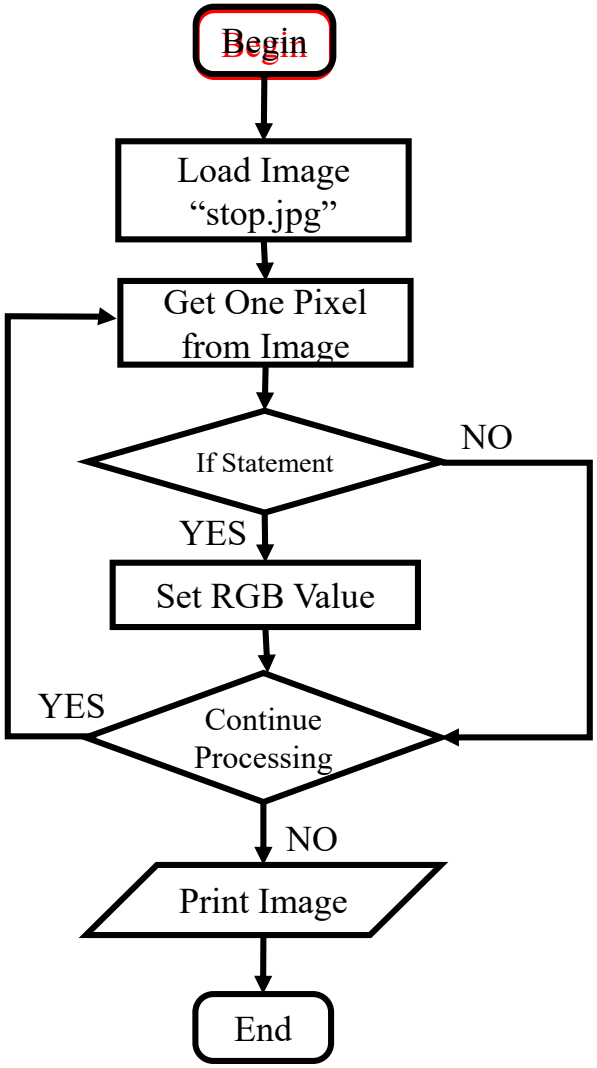
Summary

- 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?

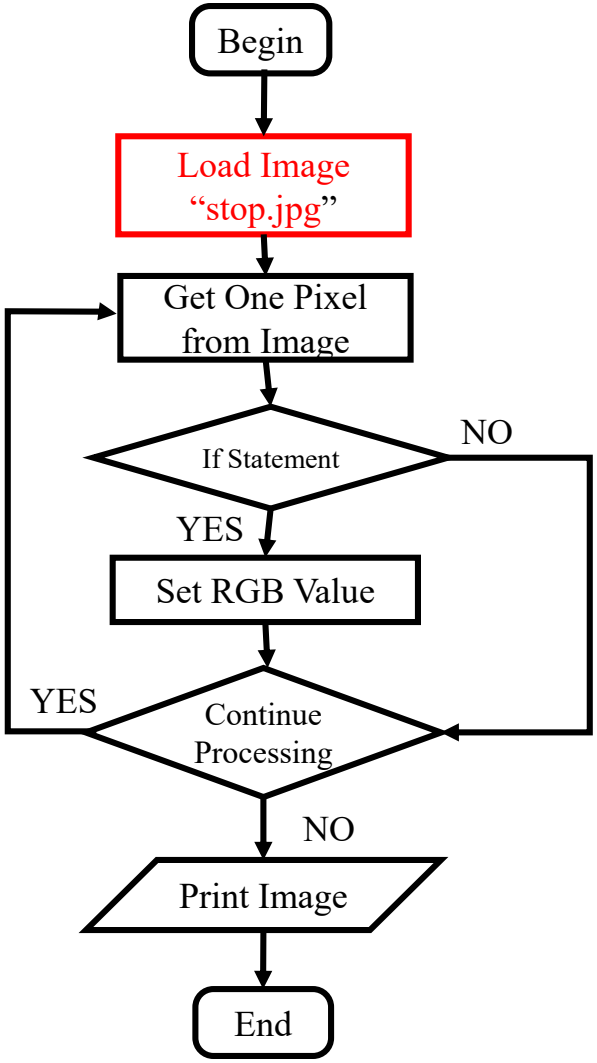
Bulk operation : Loops do some steps "for all these items"
Logic test true/false : Run some code only if a test is true

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

for (pixel : image) {
   if (pixel.getX() < 100) {
            pixel.setRed(0);
            pixel.setGreen(0);
            pixel.setBlue(255);
   }
}

print (image);
```



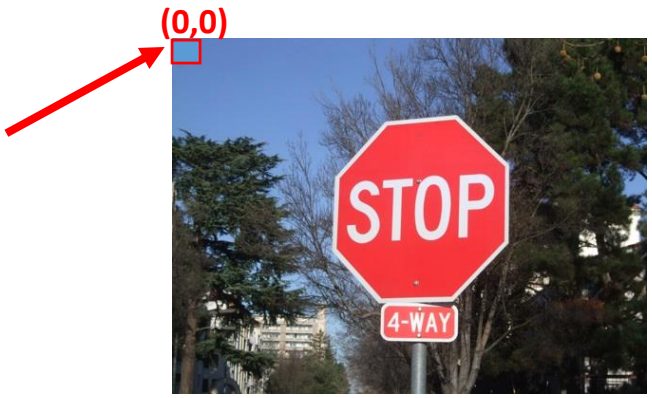
If Exercise

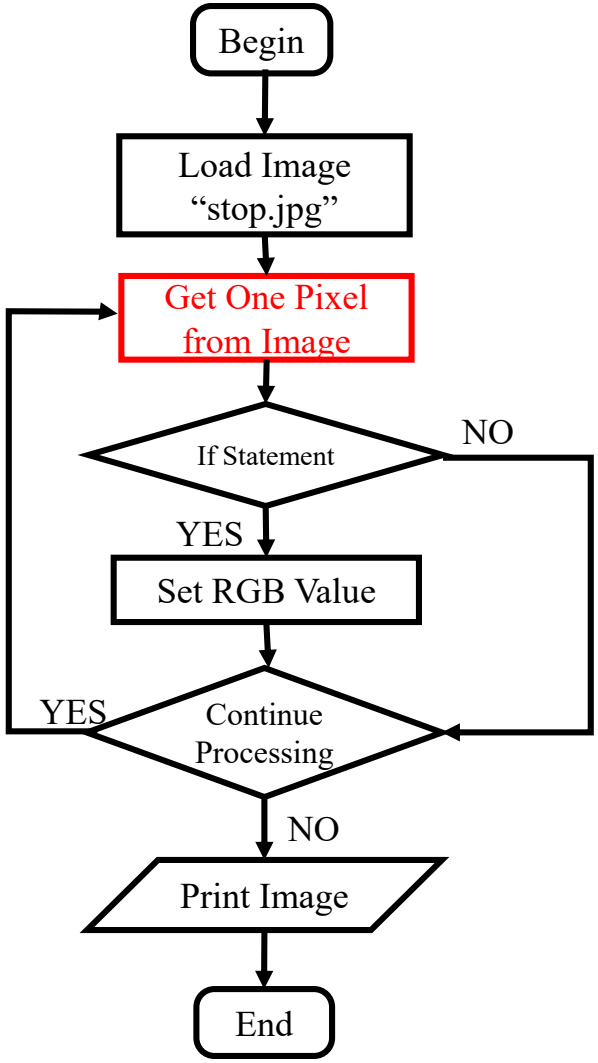
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
         pixel.setRed(0);
         pixel.setGreen(0);
         pixel.setBlue(255);
         }
}

print (image);
```



If Exercise

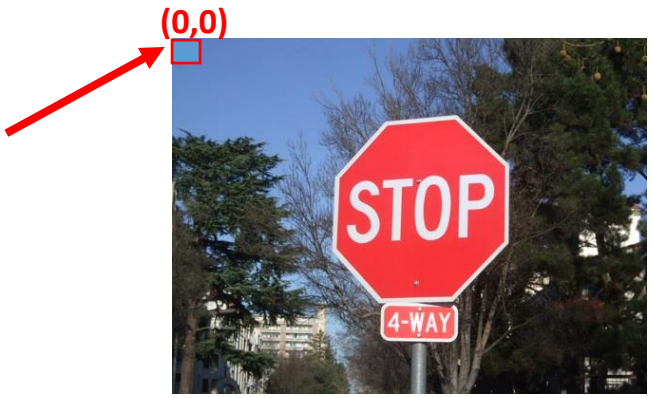
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
         pixel.setRed(0);
         pixel.setGreen(0);
         pixel.setBlue(255);    }
}

print (image);
```

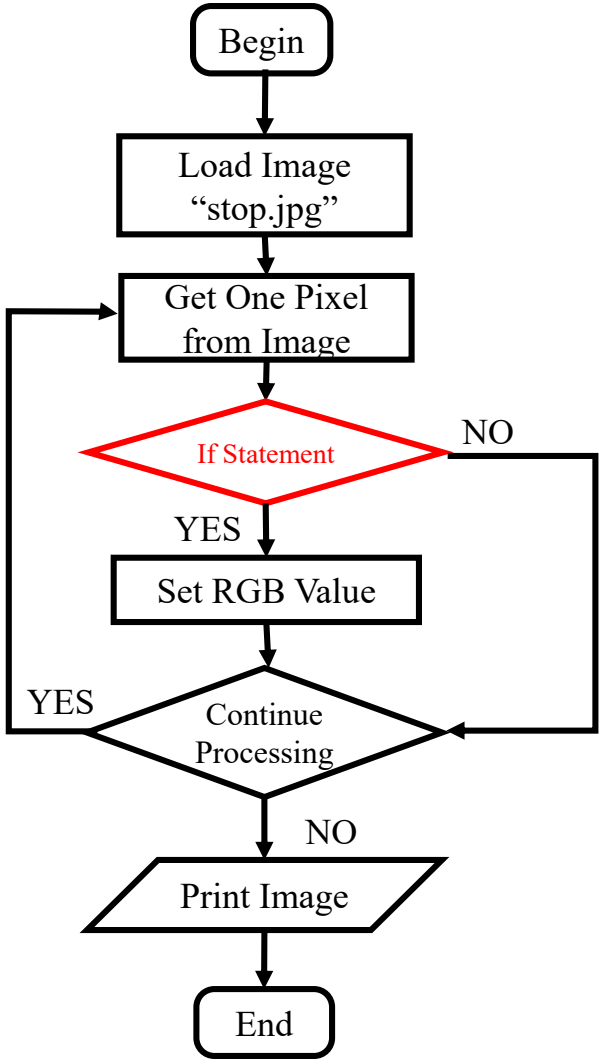




If Exercise

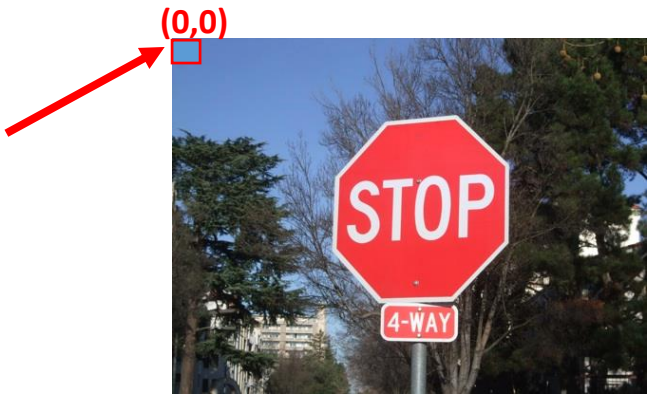
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
         pixel.setRed(0);
         pixel.setGreen(0);
         pixel.setBlue(255);    }
}

print (image);
```

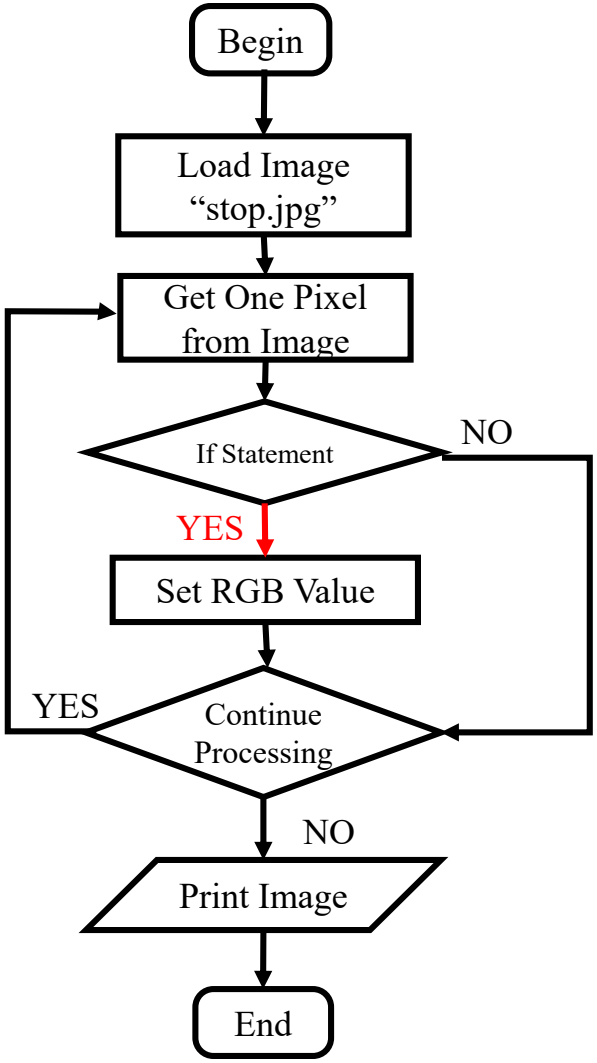




If Exercise

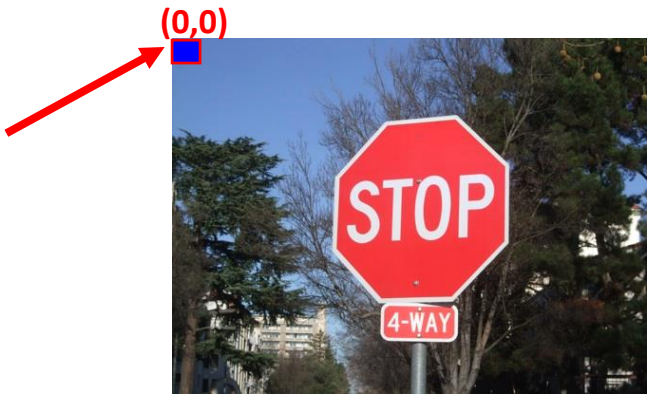
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
            pixel.setRed(0);
            pixel.setGreen(0);
            pixel.setBlue(255);
         }
}

print (image);
```

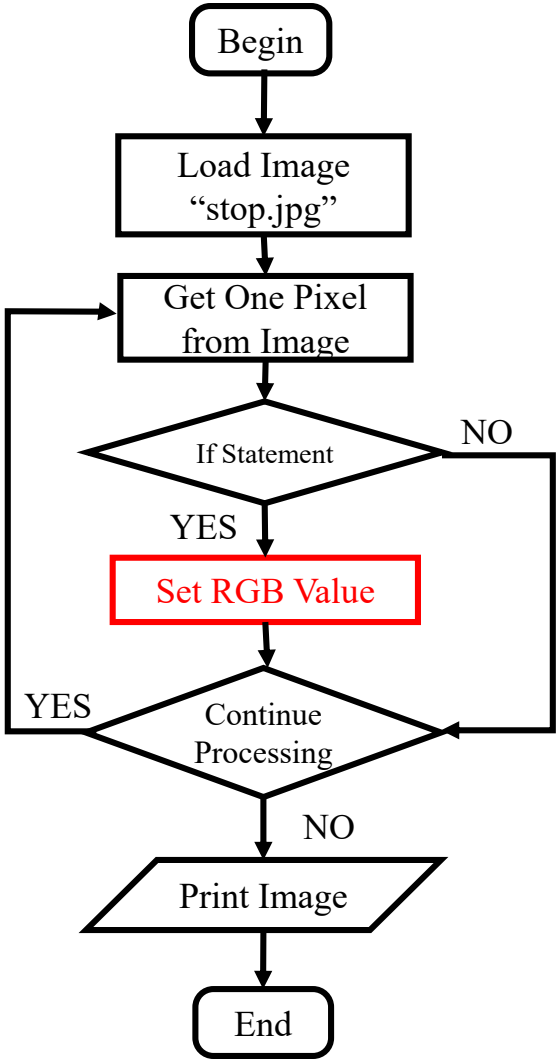




If Exercise

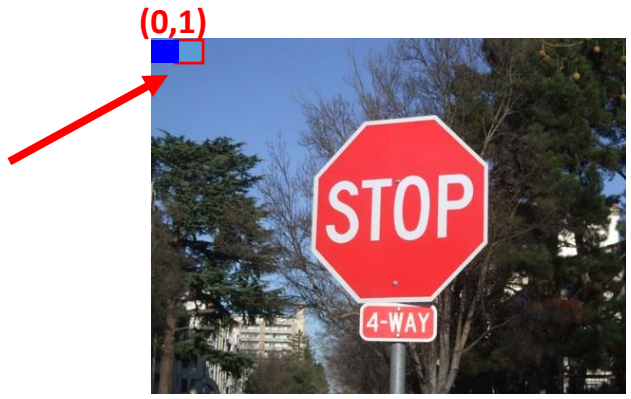
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
    if (pixel.getX() < 100) {
        pixel.setRed(0);
        pixel.setGreen(0);
        pixel.setBlue(255);
    }
}

print (image);
```

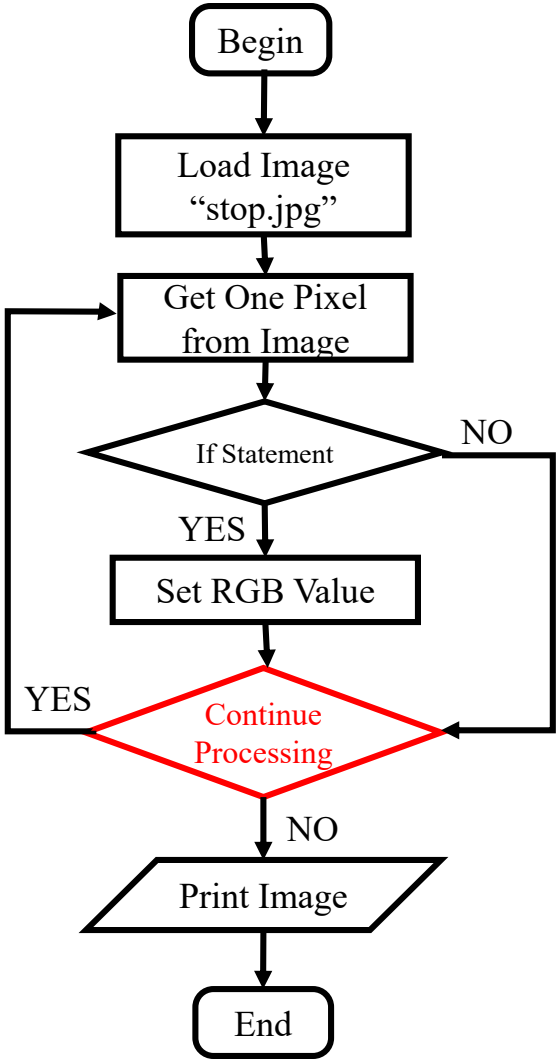




If Exercise

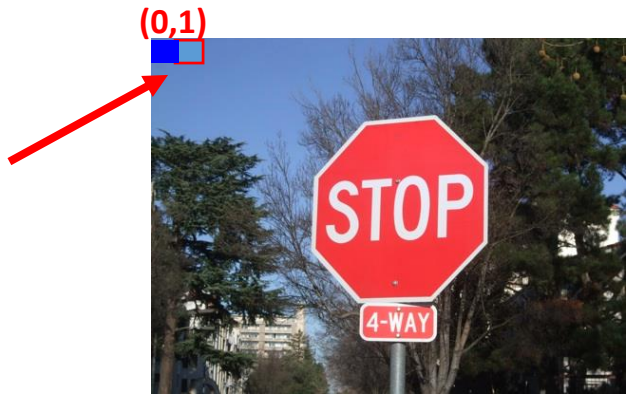
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

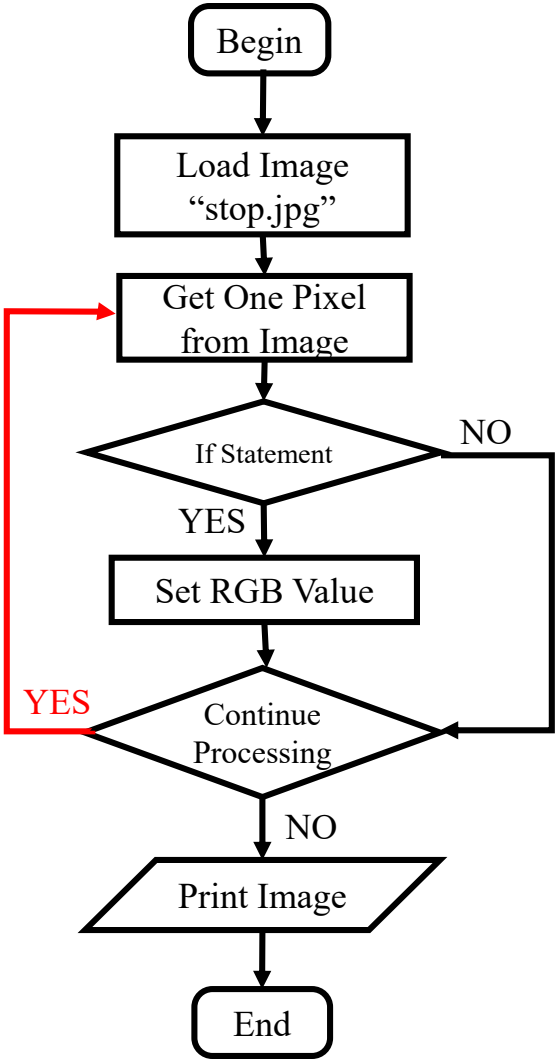




If Exercise

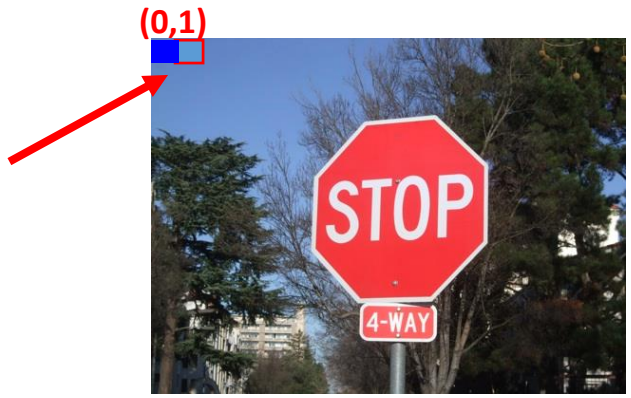
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

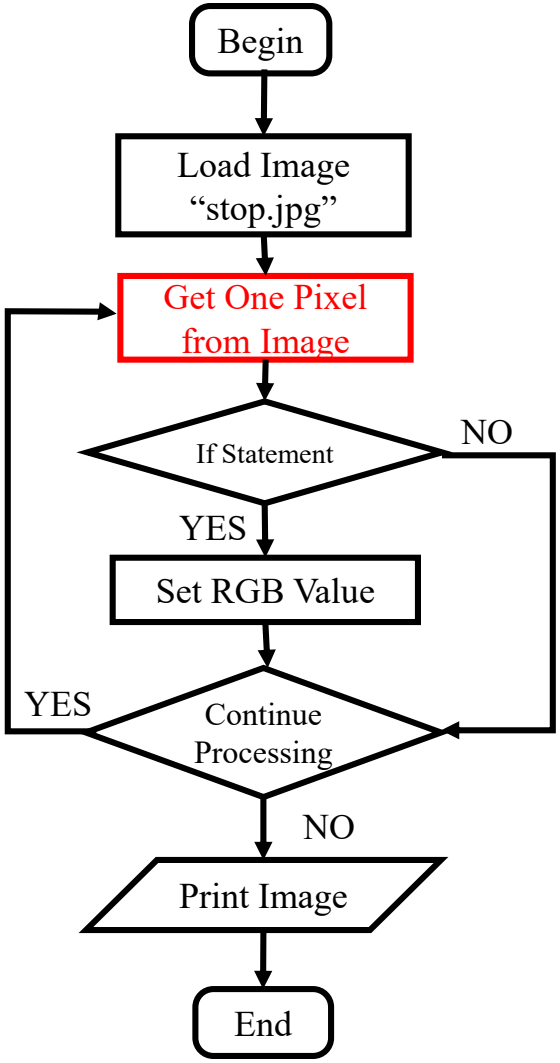




If Exercise

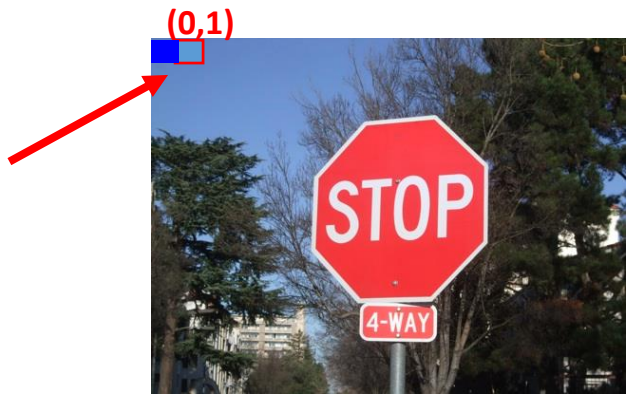
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

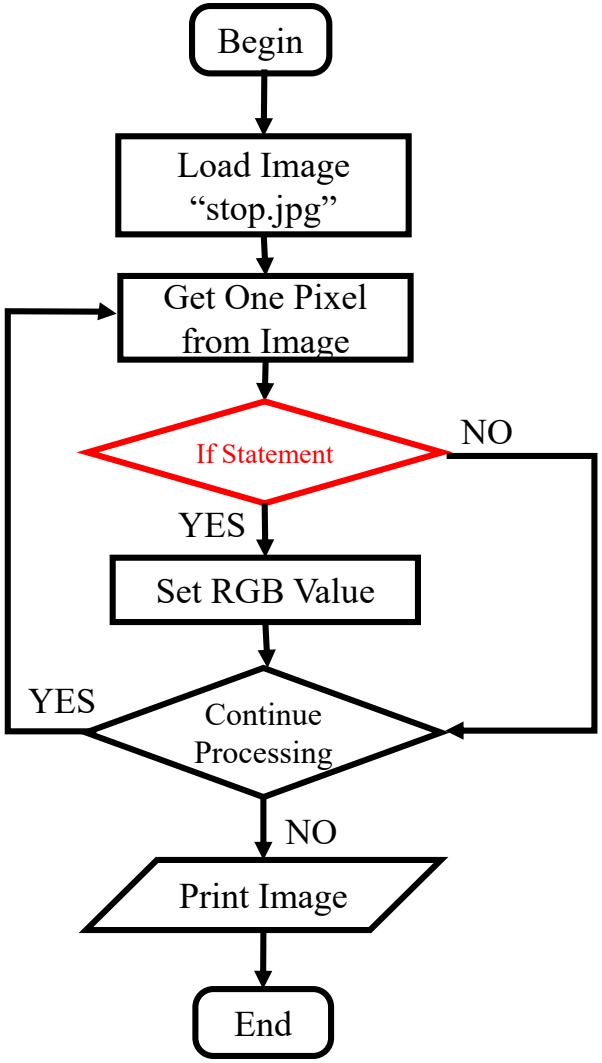




If Exercise

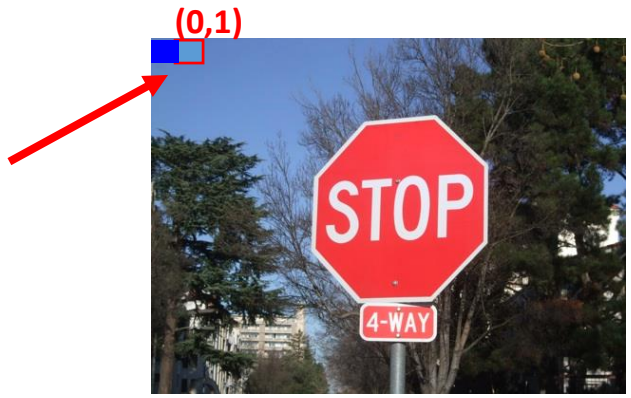
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

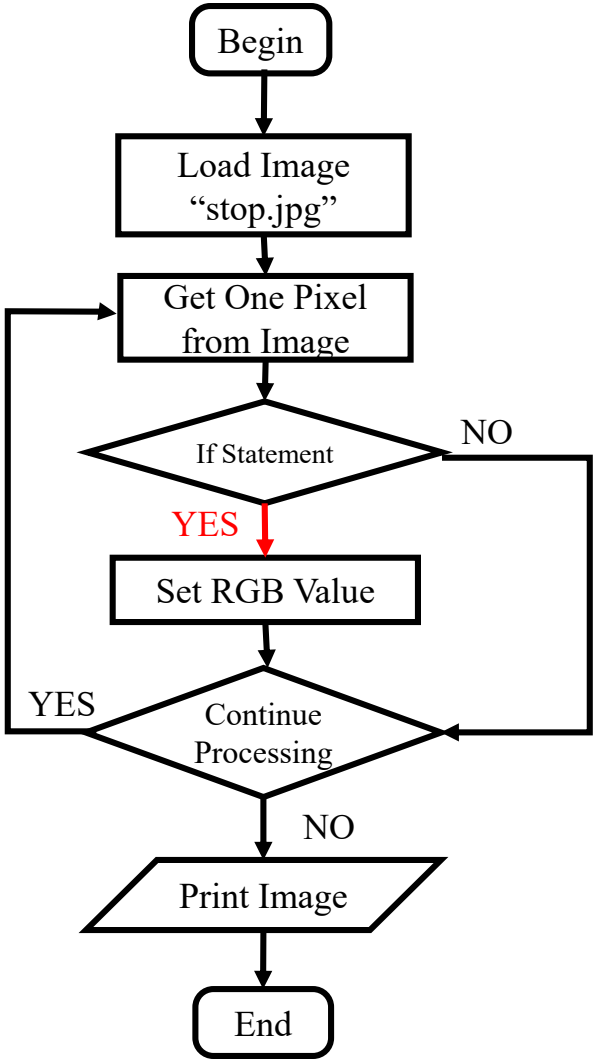




If Exercise

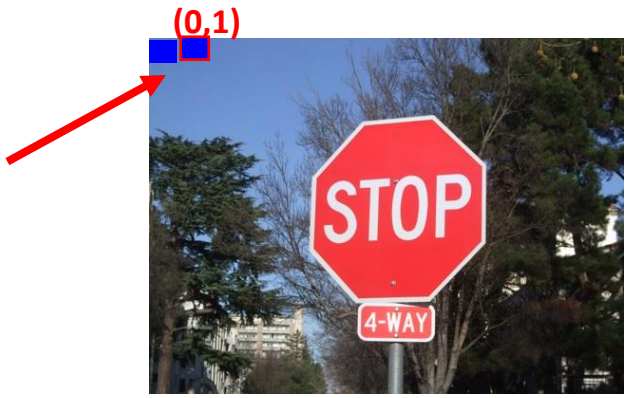
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

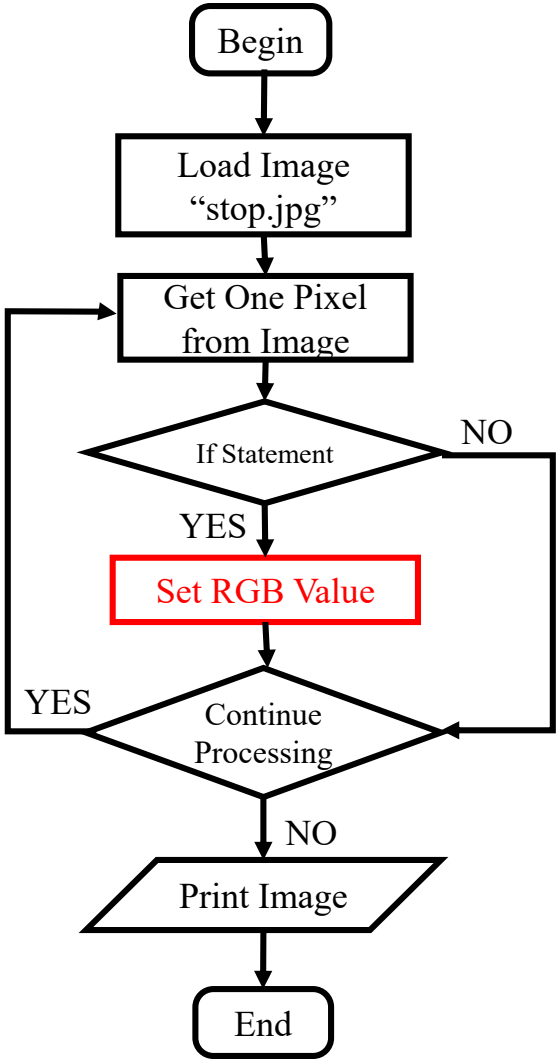




If Exercise

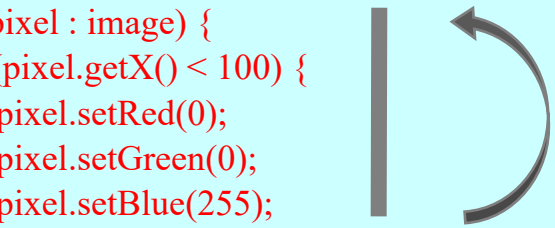
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



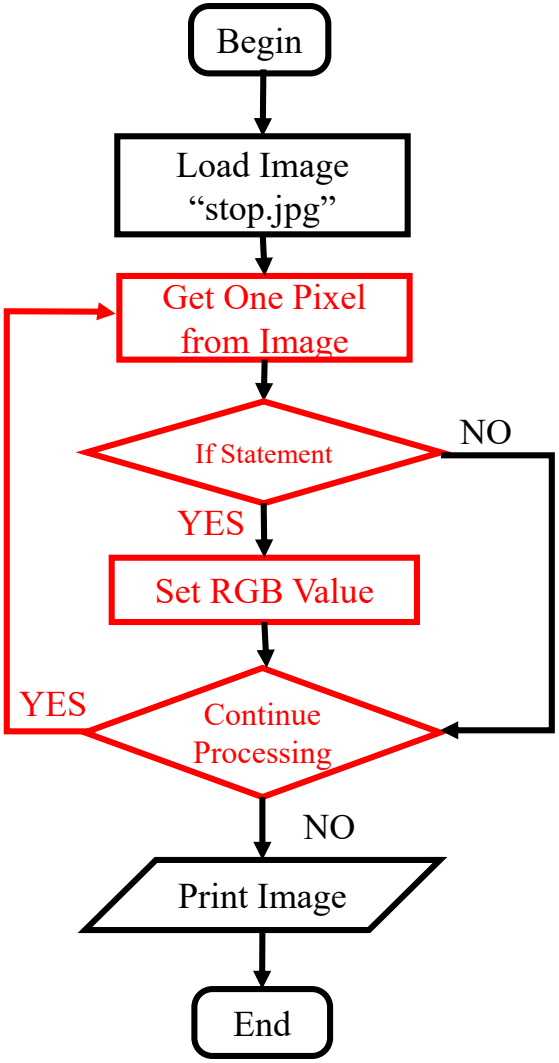


If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

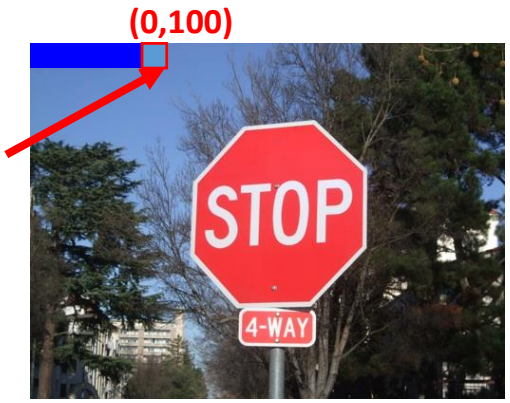


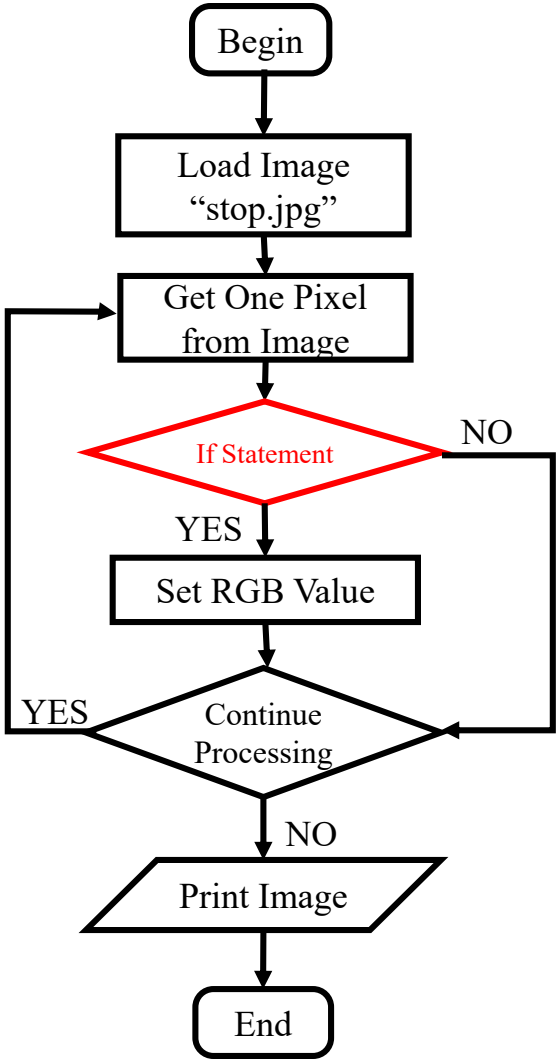
computer runs it again and again, until to （0，99）



If Exercise

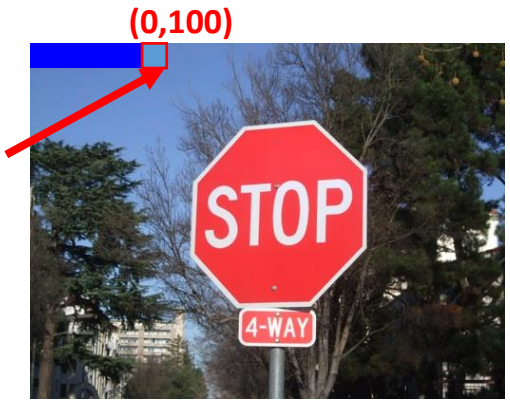
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

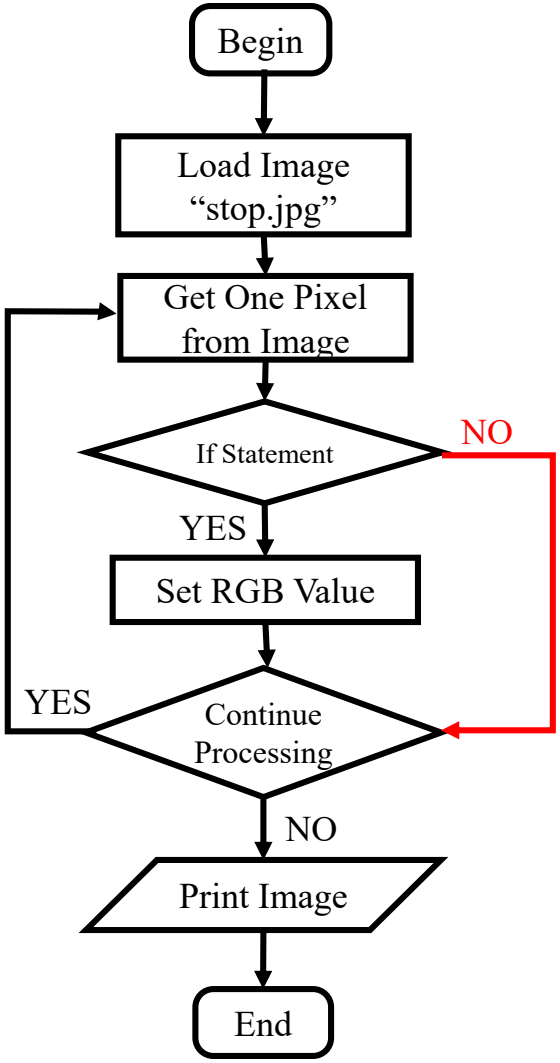




If Exercise

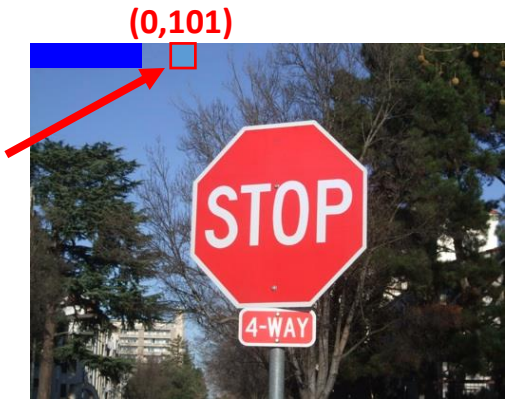
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

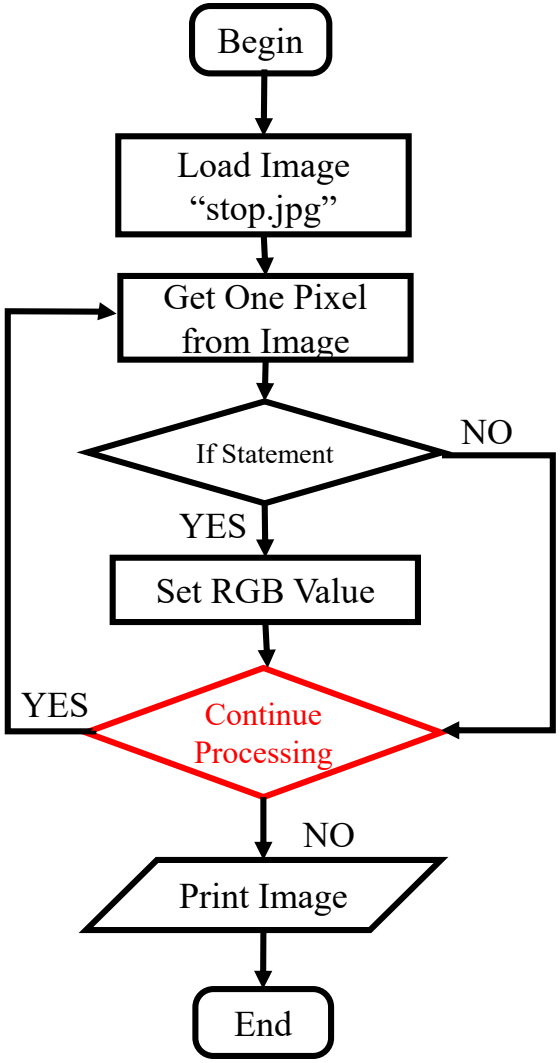




If Exercise

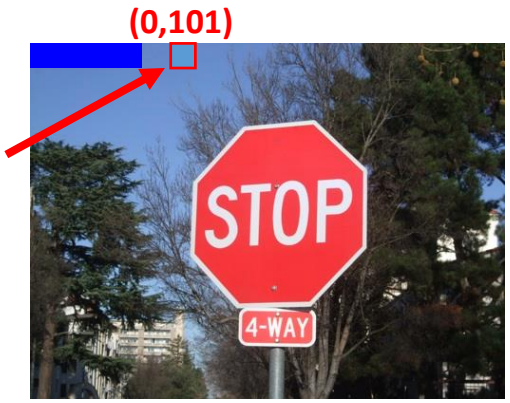
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

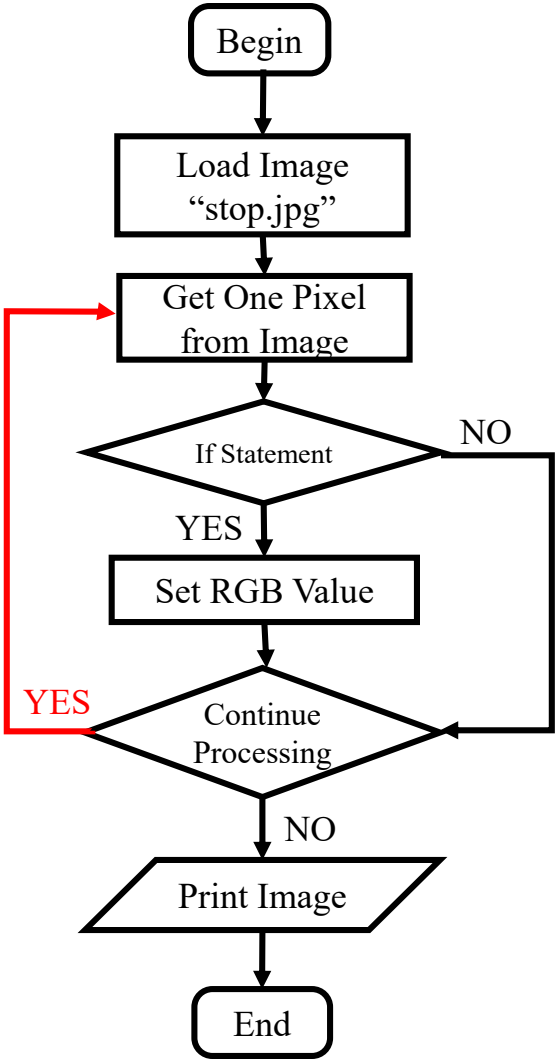




If Exercise

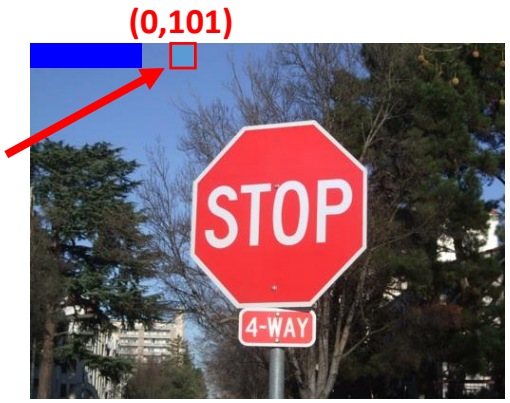
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

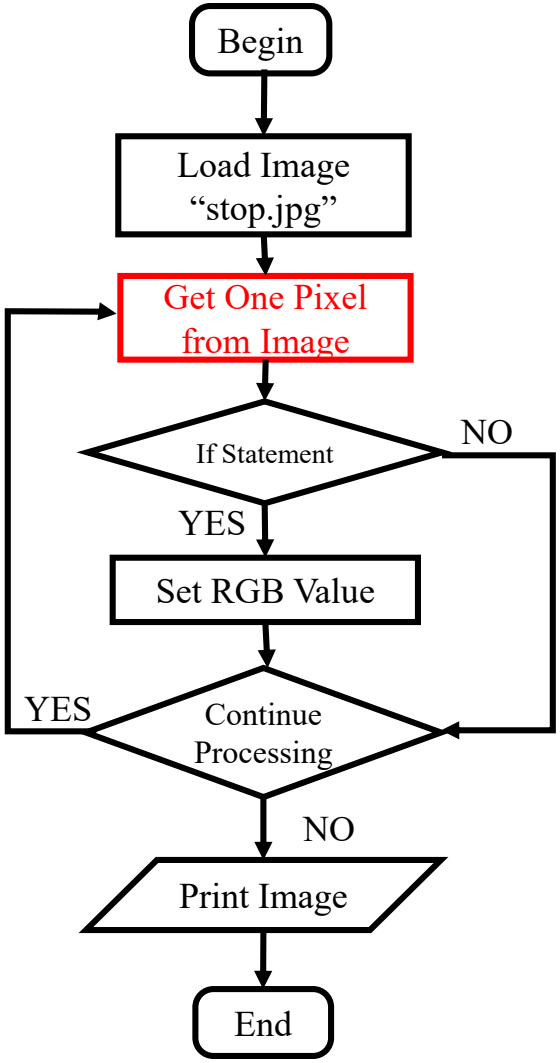




If Exercise

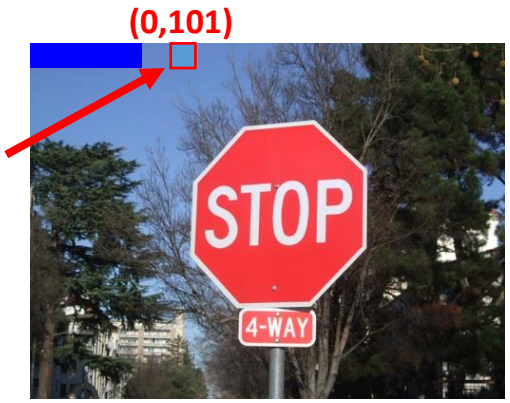
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

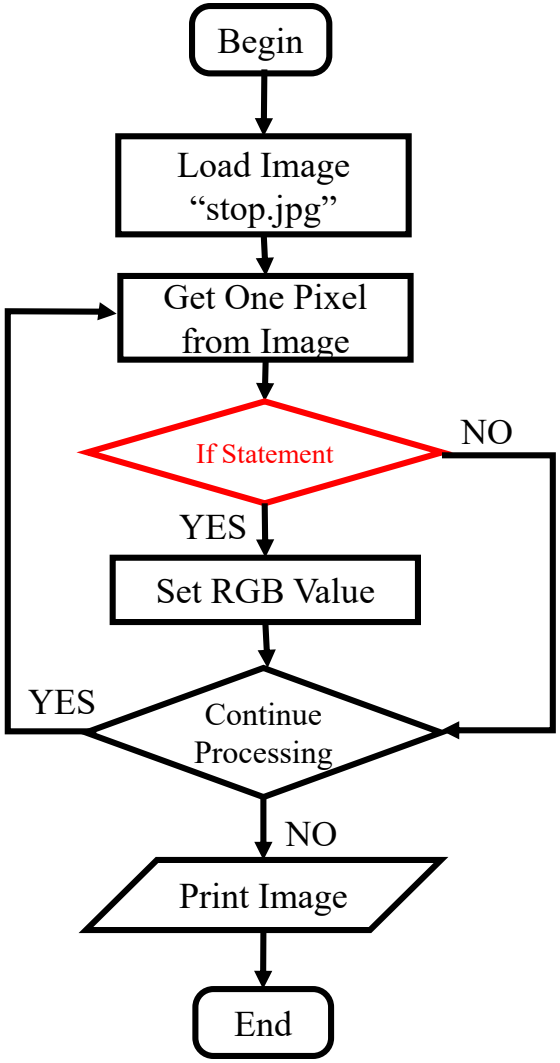




If Exercise

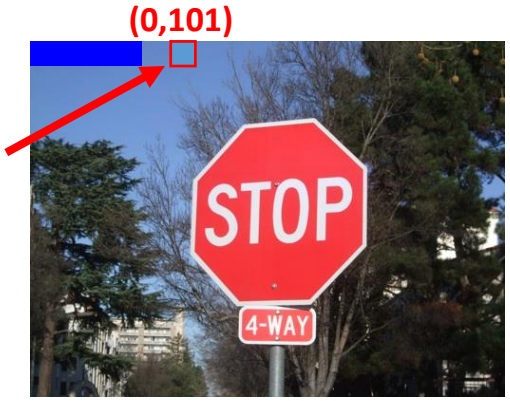
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

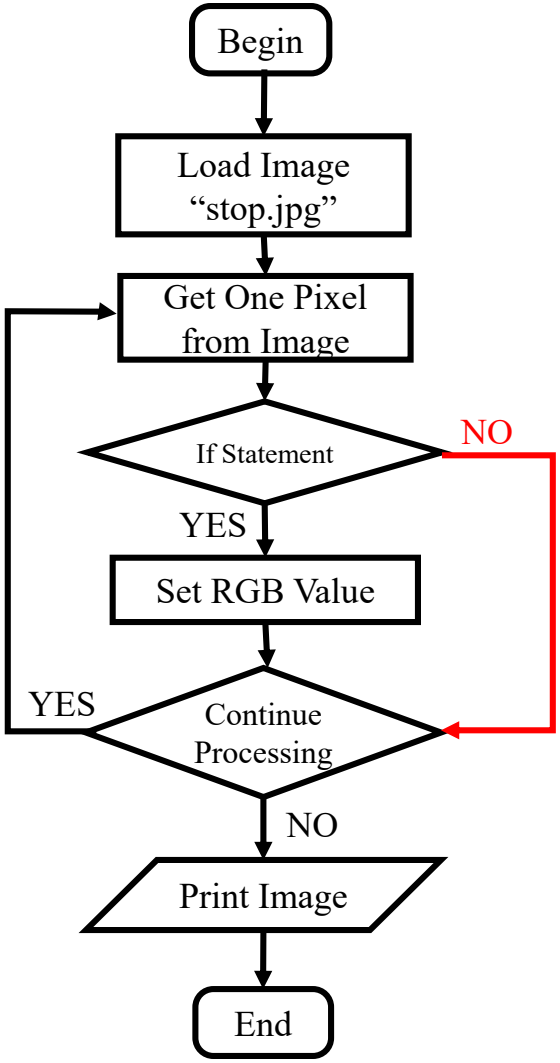




If Exercise

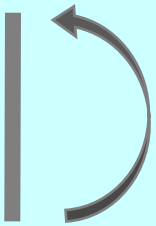
```
image = new SimpleImage("stop.jpg");

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



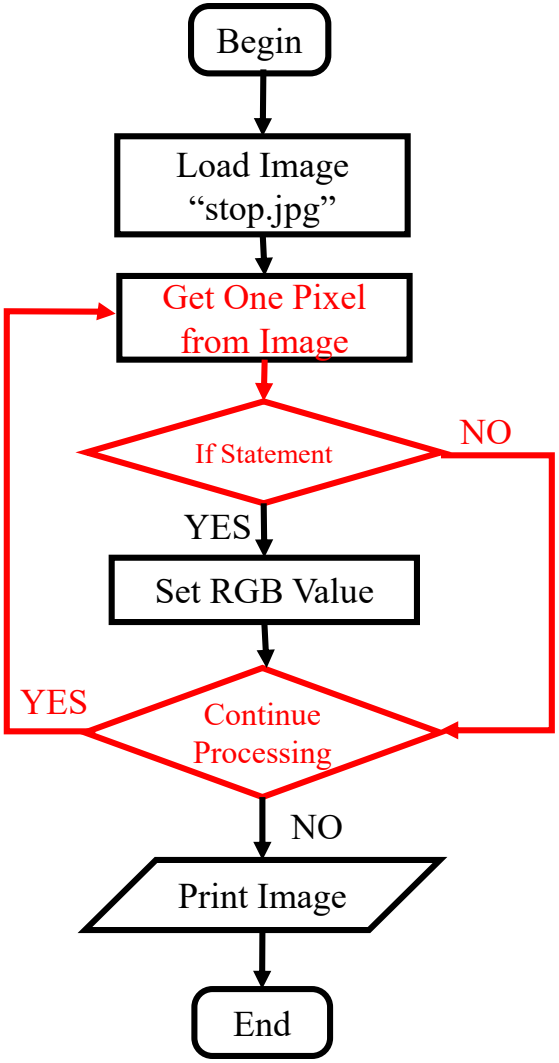


If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```

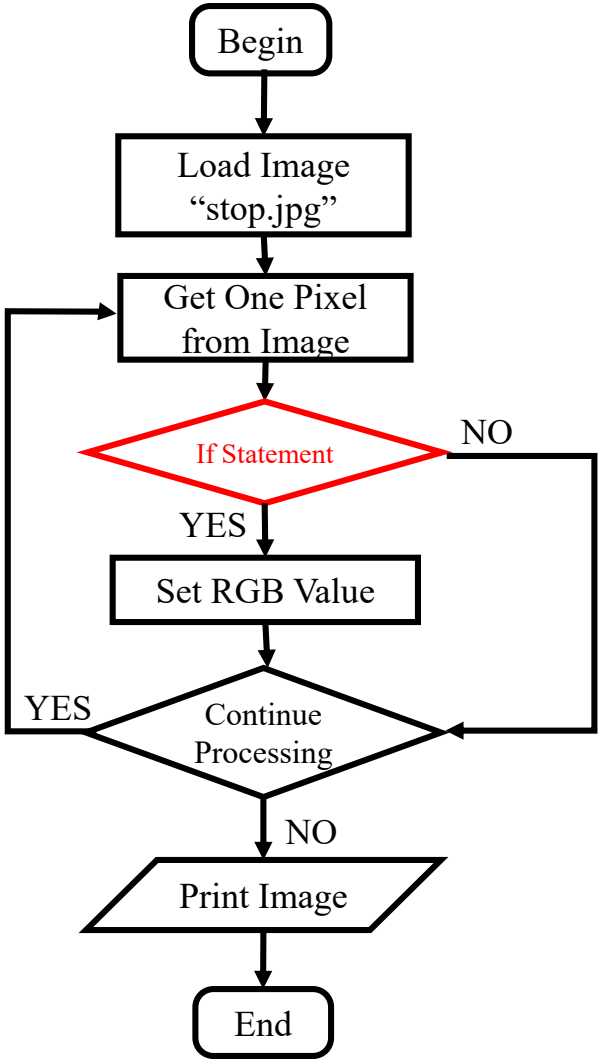


computer runs it again and again, until to （0，499）



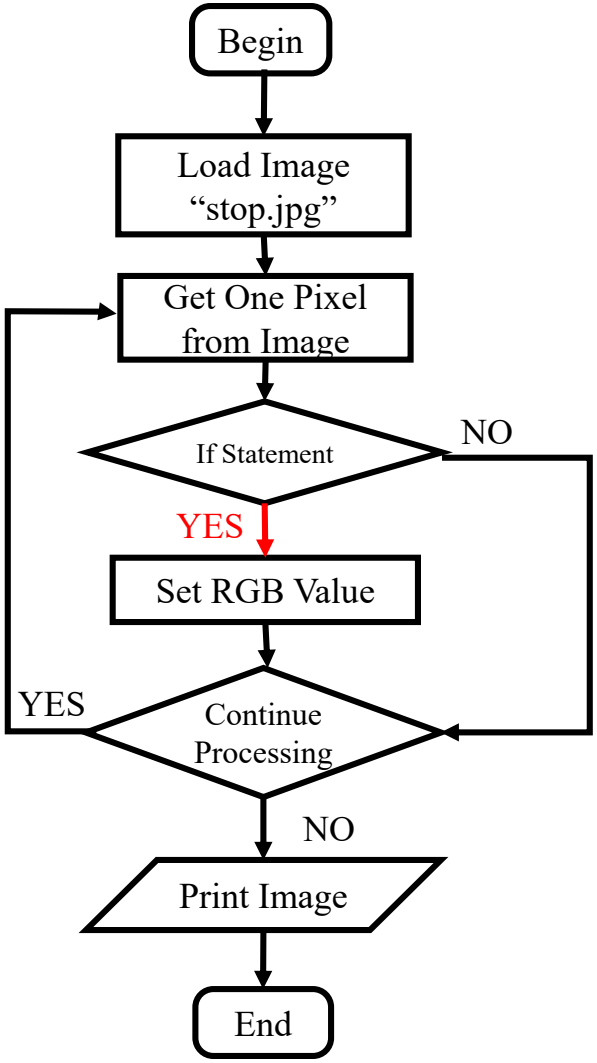
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



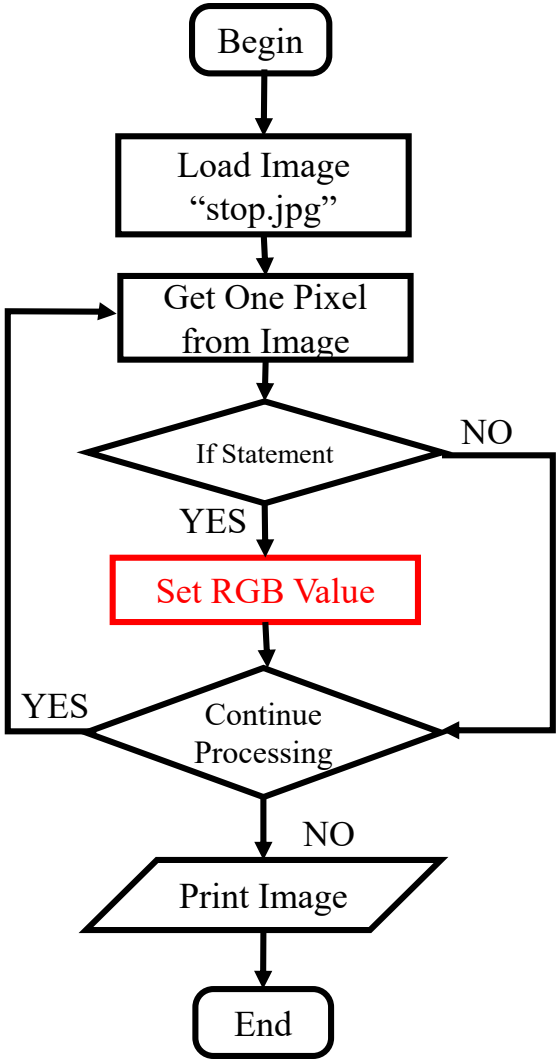
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



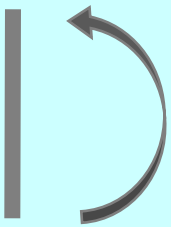
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



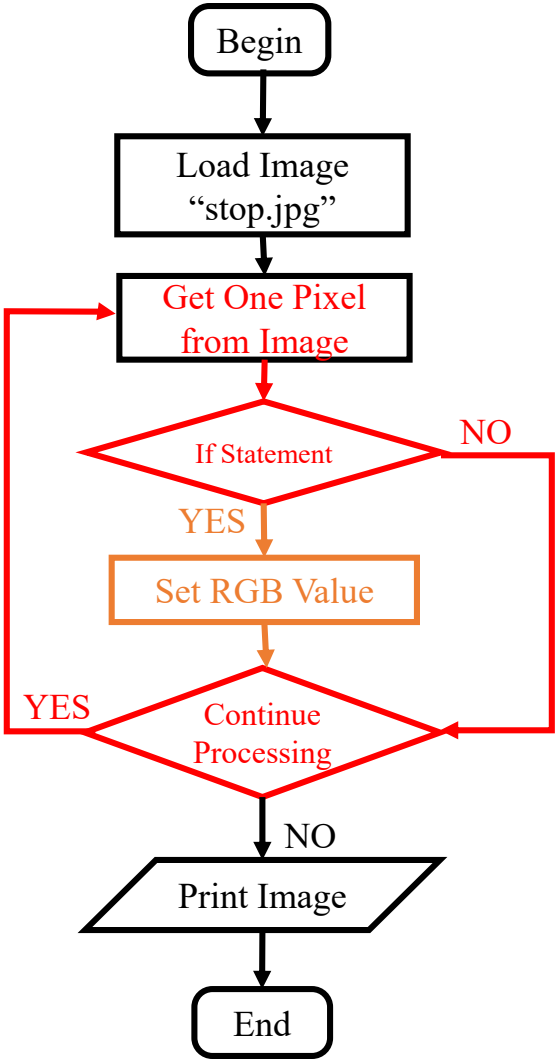
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(05);
      pixel.setBlue(255);
   }
}

print (image);
```



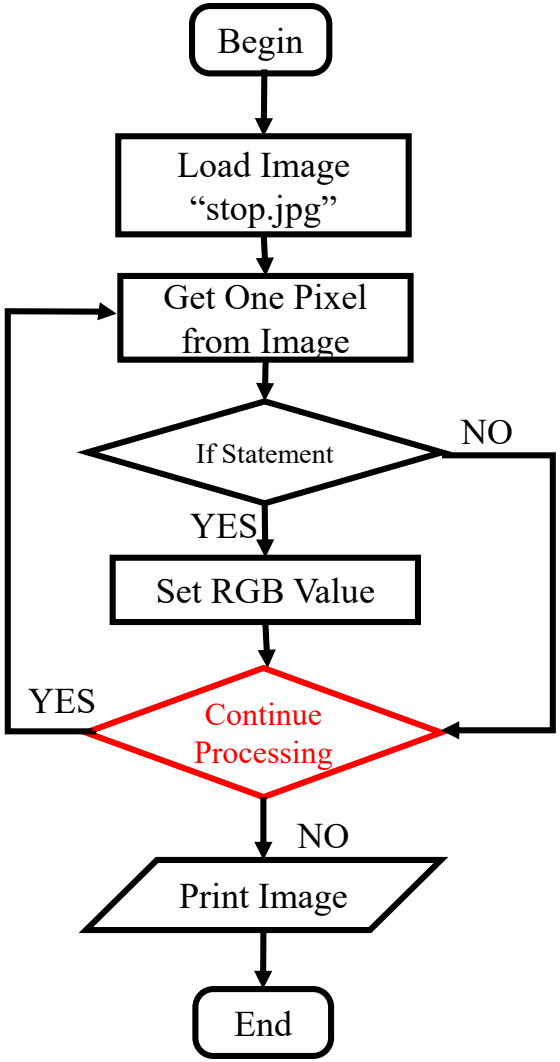
computer runs it again and again, until to （319，499）

(319,499)



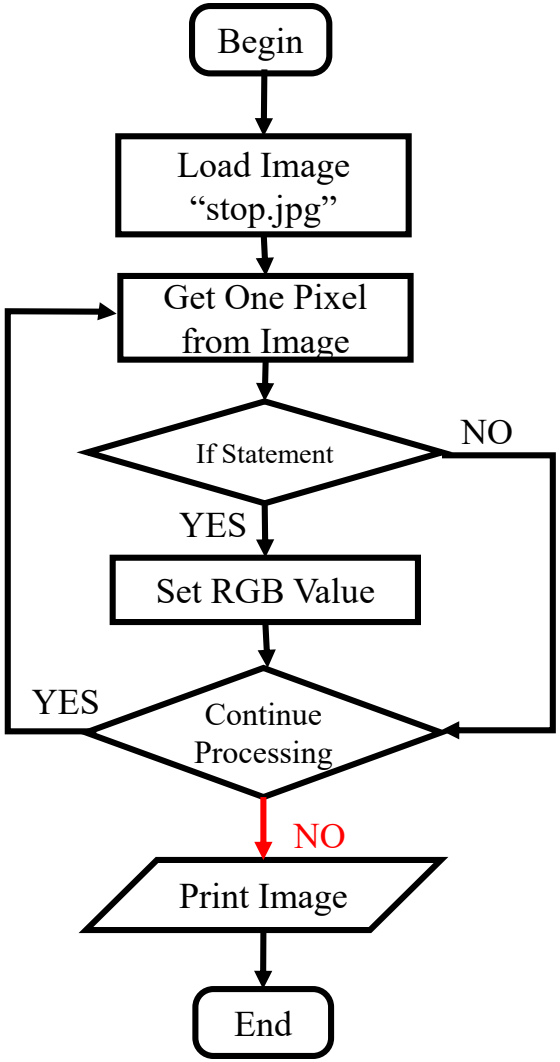
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



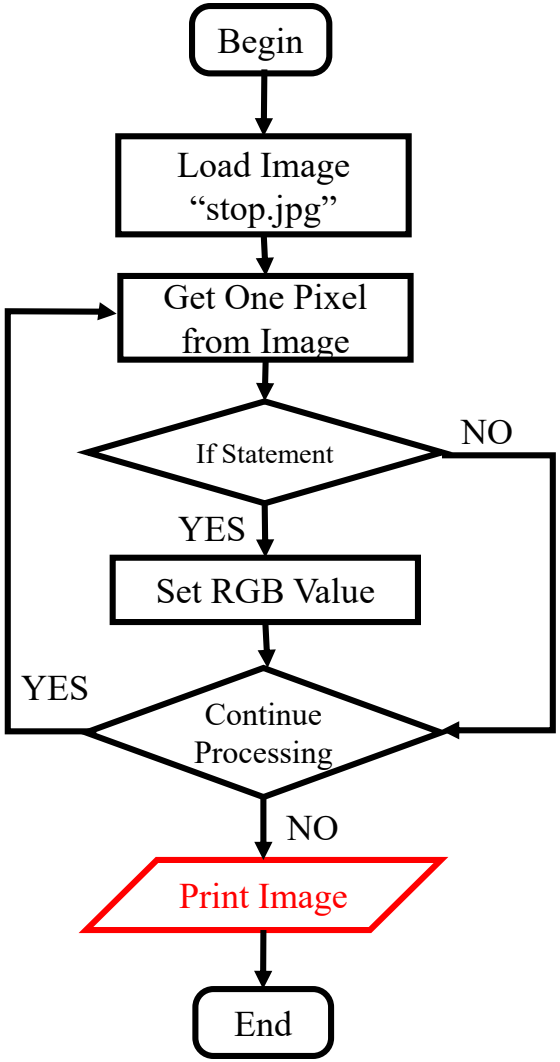
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



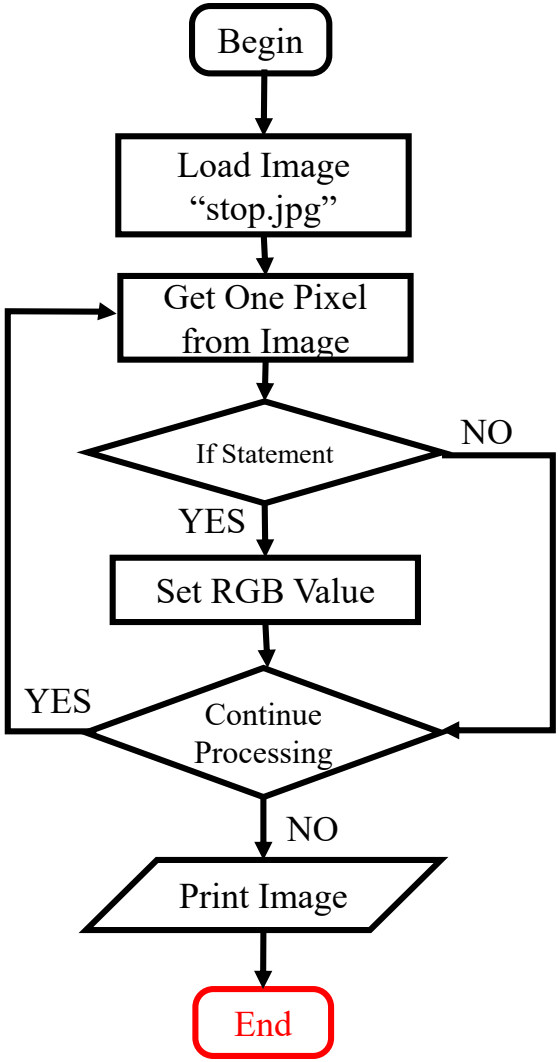
If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



If Exercise

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

for (pixel : image) {
   if (pixel.getX() < 100) {
      pixel.setRed(0);
      pixel.setGreen(0);
      pixel.setBlue(255);
   }
}

print (image);
```



If Exercise

If statement :

- if (pixel.getX() < 100) { - "test" inside parenthesis
- Curly braces around body code after the test { .. }
- 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;

If Statement Exercise

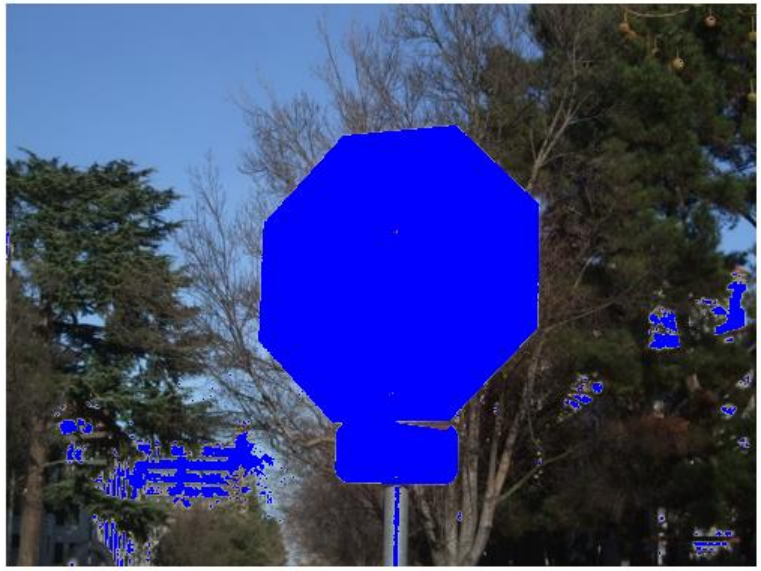
Example 1

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

How to change the red sign to be blue?

Example 2

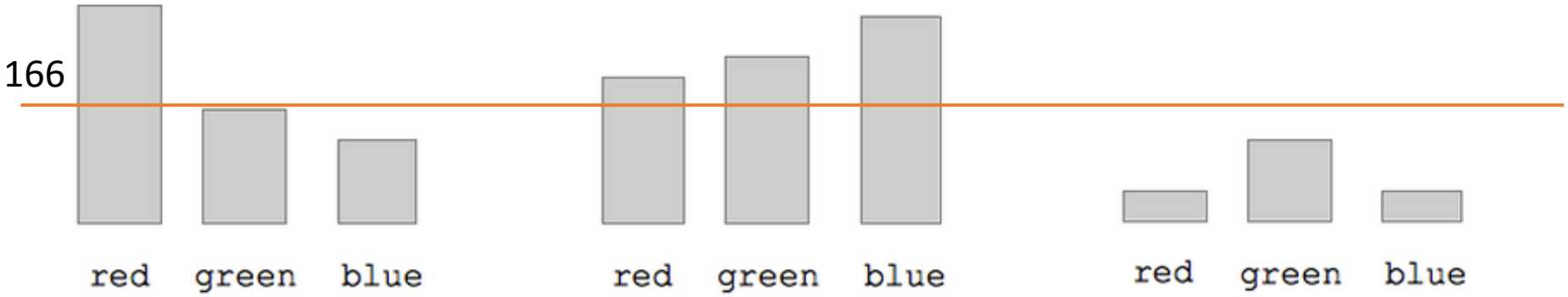
```
image = new SimpleImage("stop.jpg");
for (pixel: image) {
  if (pixel.getRed() > 160) {
    pixel.setRed(0);
    pixel.setGreen(0);
    pixel.setBlue(255);
  }
}
print(image);
```



If with Color Exercise

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:



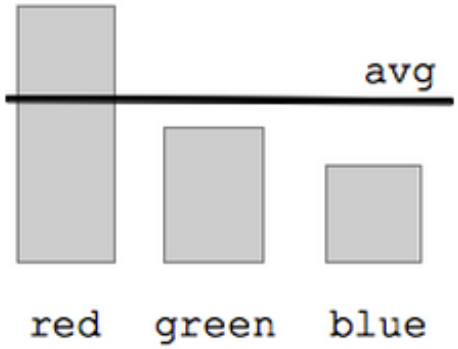
Recall Average of Pixel Color

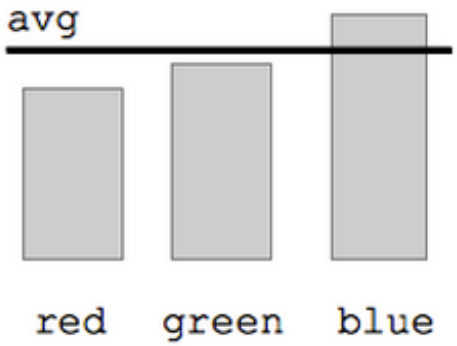
Recall this code which, inside a loop, computes the average of the red, blue, and green values and stores that number in a variable "avg"

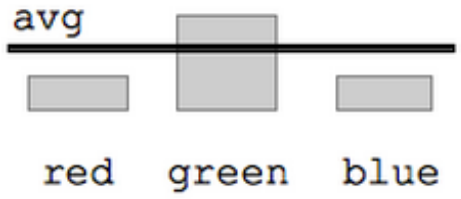
```
avg = (pixel.getRed() + pixel.getGreen() + pixel.getBlue())/3;
```

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);
```

Color Avg Test Example

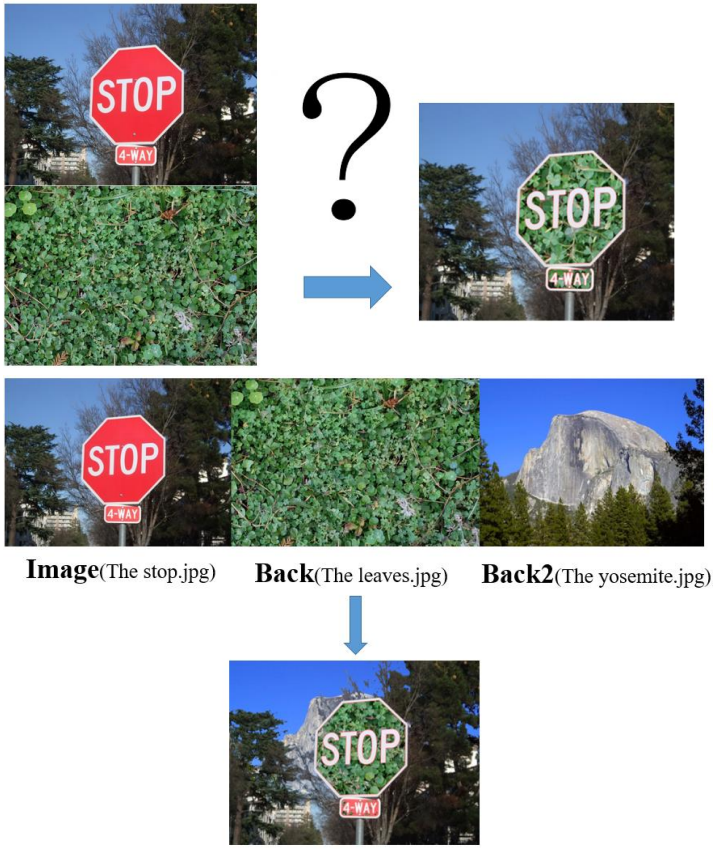
Example 4: Color Avg Test

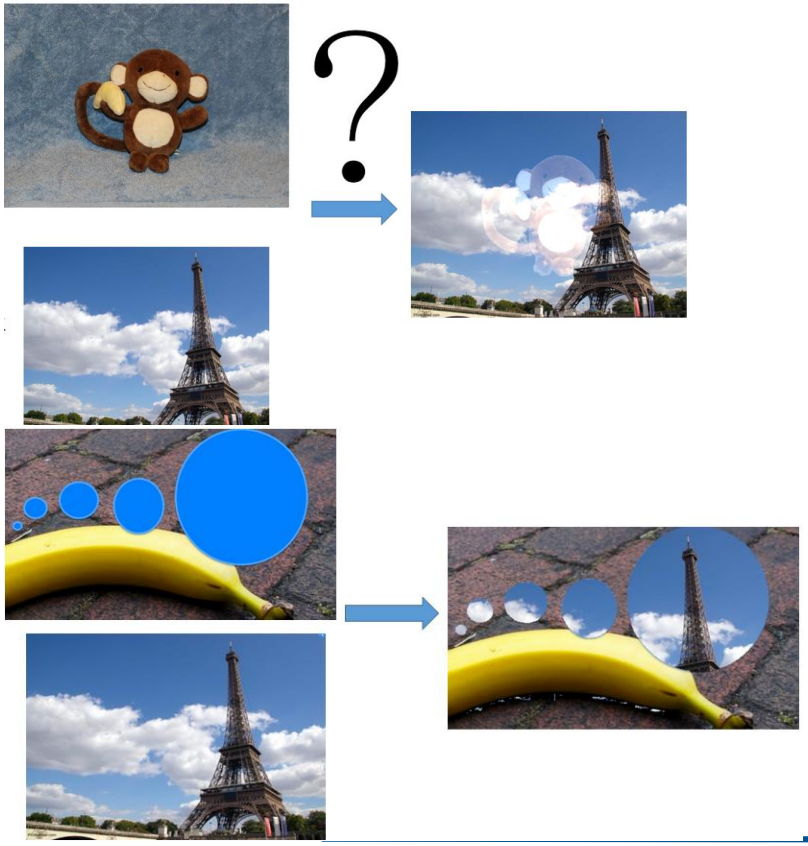
Write code to detect the red curb, tuning the 1.1 factor to look the best. Then (a) change the curb to medium gray red=120 green=120 blue=120. (b) change just the curb to be**grayscale**, which will look more realistic.

```
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 September 30;**
  - **Presentation: no more than 25 minutes;**
  - **Try your best, and show your bests.**

Thank You!

Q&A