

Chapter 5: Thinking Like a Programmer

One of the hardest things to learn is how to think like a programmer. A programmer is not created by simple books or classes but grows from within an individual. To become a "good" programmer takes passion for technology, self learning, basic intelligence, and a drive to create and explore.

You are like the great explorers Christopher Columbus, Neil Armstrong, and Yuri Gagarin (the first human in space). You have an unlimited universe to explore and to create within the computer. The only restrictions on where you can go will be your creativity and willingness to learn.

A program to develop a game or interesting application can often exceed several thousand lines of computer code. This can very quickly become overwhelming, even to the most experienced programmer. Often we programmers will approach a complex problem using a three step process, like:

1. Think about the problem.
2. Break the problem up into pieces and write them down formally.
3. Convert the pieces into the computer language you are using.

Pseudocode:

Pseudocode is a fancy word for writing out, step by step, what your program needs to be doing. The word pseudocode comes from the Greek prefix "pseudo-" meaning fake and "code" for the actual computer programming statements. It is not created for the computer to use directly but it is made to help you understand the complexity of a problem and to break it down into meaningful pieces.

There is no single best way to write pseudocode. Dozens of standards exist and each one of them is very suited for a particular type of problem. In this

introduction we will use simple English statements to understand our problems.

How would you go about writing a simple program to draw a school bus (like in Illustration 13)?

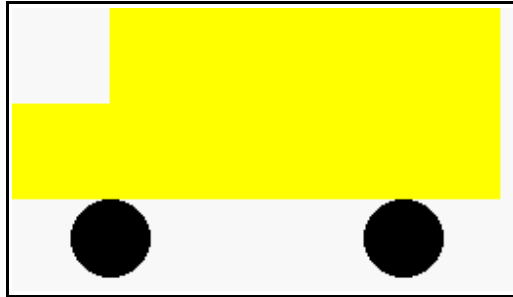


Illustration 13: School Bus

Let's break this problem into two steps:

- draw the wheels
- draw the body

Now let's break the initial steps into smaller pieces and write our pseudocode:

```
Set color to black.  
Draw both wheels.  
Set color to yellow.  
Draw body of bus.  
Draw the front of bus.
```

Table 4: School Bus - Pseudocode

Now that we have our program worked out, all we need to do is write it:

Set color to black.	<code>color black</code>
Draw both wheels.	<code>circle 50,120,20</code> <code>circle 200,120,20</code>
Set color to yellow.	<code>color yellow</code>
Draw body of bus.	<code>rect 50,0,200,100</code>
Draw the front of bus.	<code>rect 0,50,50,50</code>

Table 5: School Bus - Pseudocode with BASIC-256 Statements

The completed school bus program (Program 28) is listed below. Look at the finished program and you will see comment statements used in the program to help the programmer remember the steps that they used to initially solve the problem.

```

1 # schoolbus.kbs
2 # draw a school bus
3
4 clg
5
6 # draw wheels
7 color black
8 circle 50,120,20
9 circle 200,120,20
10
11 # draw bus body
12 color yellow
13 rect 50,0,200,100
14 rect 0,50,50,50

```

Program 28: School Bus

In the school bus example we have just seen there were many ways to break up the problem. You could have drawn the bus first and the wheels last, you

could have drawn the front before the back,... We could list dozens of different ways this simple problem could have been tackled.

One very important thing to remember, THERE IS NO WRONG WAY to approach a problem. Some ways are better than others (fewer instructions, easier to read, ...), but the important thing is that you solved the problem.

Flowcharting:

Another technique that programmers use to understand a problem is called flowcharting. Following the old adage of "a picture is worth a thousand words", programmers will sometimes draw a diagram representing the logic of a program. Flowcharting is one of the oldest and commonly used methods of drawing what a program is supposed to do.

This brief introduction to flowcharts will only cover a small part of what can be done with them, but with a few simple symbols and connectors you will be able to model very complex processes. This technique will serve you well not only in programming but in solving many problems that you will come across. Here are a few of the basic symbols:


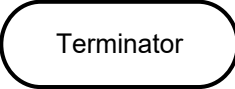

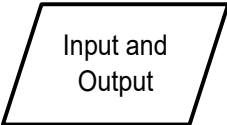
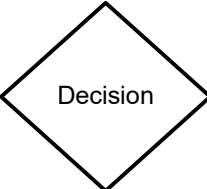
Symbol	Name and Description
	Flow – An arrow represents moving from one symbol or step in the process to another. You must follow the direction of the arrowhead.
	Terminator – This symbol tells us where to start and finish the flowchart. Each flowchart should have two of these: a start and a finish.
	Process – This symbol represents activities or actions that the program will need to take. There should be only one arrow leaving a process.
	Input and Output (I/O) – This symbol represents data or items being read by the system or being written out of the system. An example would be saving or loading files.
	Decision – The decision diamond asks a simple yes/no or true/false question. There should be two arrows that leave a decision. Depending on the result of the question we will follow one path out of the diamond.

Table 6: Essential Flowcharting Symbols

The best way to learn to flowchart is to look at some examples and to try your own hand at it.

Flowcharting Example One:

You just rolled out of bed and your mom has given you two choices for breakfast. You can have your favorite cold cereal or a scrambled egg. If you do not choose one of those options you can go to school hungry.

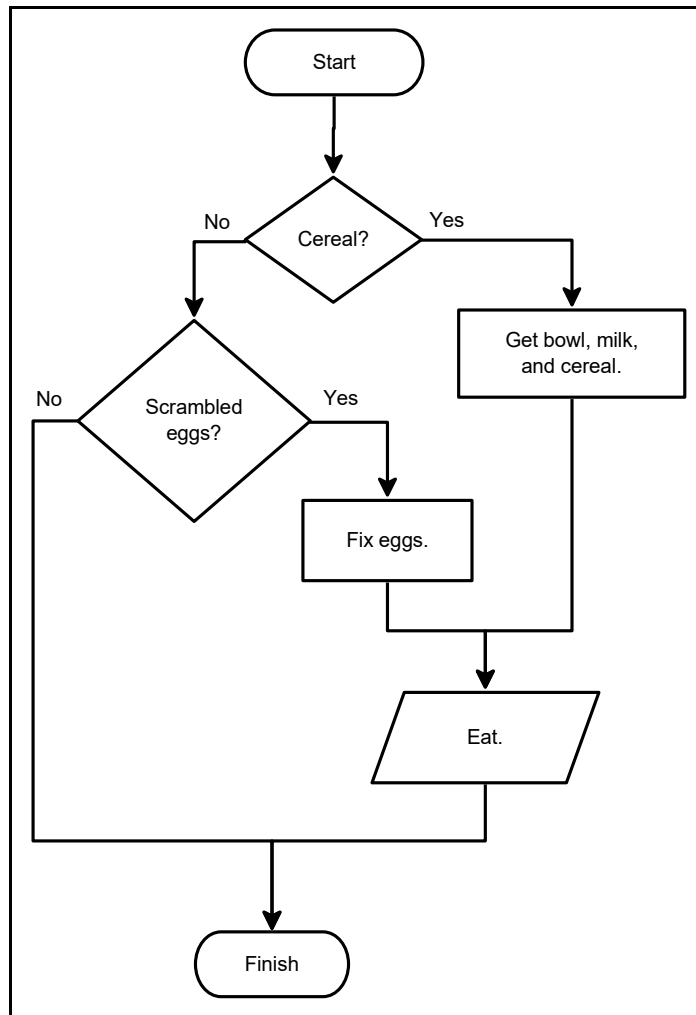


Illustration 14: Breakfast - Flowchart

Take a look at Illustration 14 (above) and follow all the arrows. Do you see how that picture represents the scenario?

Flowcharting Example Two:

Another food example. You are thirsty and want a soda from the machine. Take a look at Illustration 15 (below).

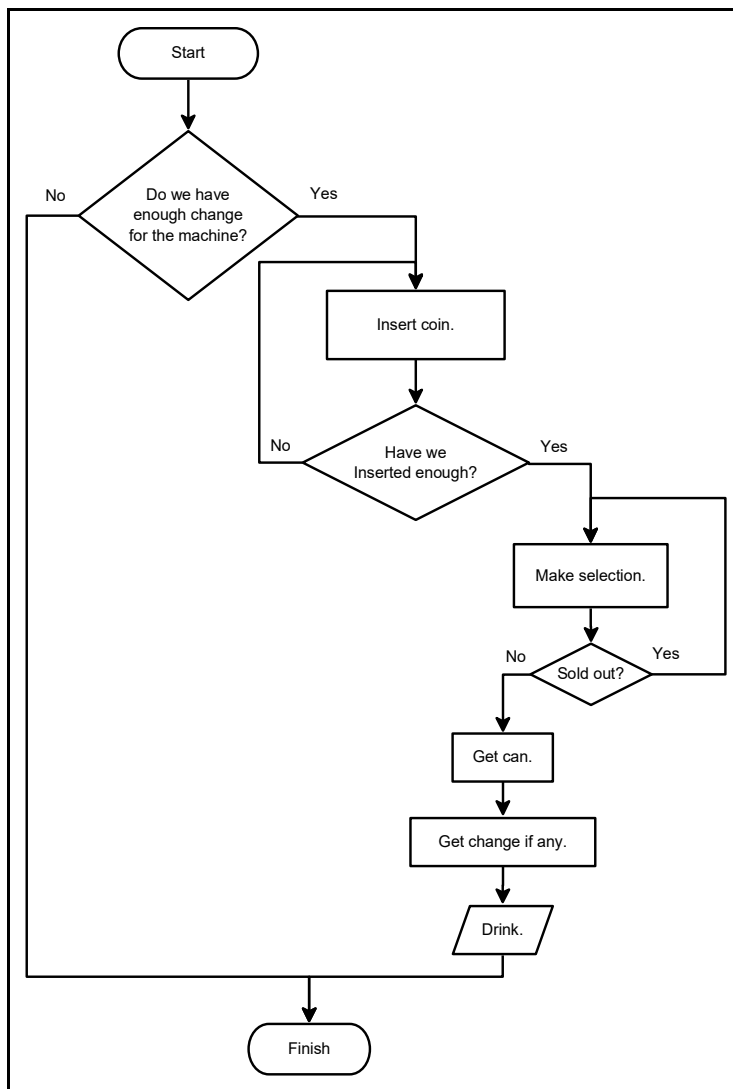




Illustration 15: Soda Machine - Flowchart

Notice in the second flowchart that there are a couple of times that we may need to repeat a process. You have not seen how to do that in BASIC-256, but it will be covered in the next few chapters.

Free eBook Edition

Exercises:

 <p>Word Search</p>	<pre> z d s y m b o l t r p e m e w t a f r m r t y d k c l u a v o s e p q o z i h p g r p r x r i c c s r n r e m z f o w o a i e i t i a u o c m d x o u s n q l h m e p u p n q a f o i q m s t e d u t b n m h r u s w s b o g e p r o b l e m p r </pre> <p>decision, flowchart, input, output, problem, process, programming, pseudocode, steps, symbol, terminator</p>
---	--

 <p>Problems</p>	<ol style="list-style-type: none"> 1. In complete sentences can you write out the steps to make a peanut butter and jelly sandwich. Assume that the peanut butter jar, jelly jar, loaf of bread, place, and silverware are on the table in front of you. Can another person, who has never seen a PBJ, successfully make one using your directions? 2. In a flow chart (or in a similar diagram) diagram the process you go through to open the front door of your house or apartment. Do you have your keys? Is the door locked? Is it already open? 3. In pseudocode (short statements) can you write out directions from your school or work to the nearest restaurant or gas station. Don't cheat and look the directions up on-line. Will the same directions get you back the same way or do the instructions need to be changed?
---	--