TABLE 4.

Semantic record label	**Address**	**Code**	**Expression translated**
T6	3		The variable Y.
T7		Load 0,A Load 2,B Mult A,B,4 Load 4,A Load 1,B Add A,B,5 Load 5,A Store A,3	The assignment statement Y = 3 + 2*X.

As you can imagine, for a large program the number and complexity of the semantic records may become immense. The compiler must have rules for all types of expressions, control structures like if and for, function calls and much more. But, compilation is not a magic process; each step can be written down and straightforwardly explained.

Exercises for 8.1.3

1. Give a parse tree with semantic records for the statement `Z = 2 + 2.` Show the assembly language translation that results.
2. Give a parse tree with semantic records for the statement `C = 9 * 4 * 5`. Show the assembly language translation that results.
3. Why is a symbol table needed? Why is it that a constant or variable cannot be given an address immediately, when its semantic record is created?
4. Give a set of semantic rules for generating semantic records. Your set should have seven rules to correspond with the seven syntax rules.

8.1.4 Grammars for Fractal trees

Formal grammars have not just been applied to natural and computer languages. They have a wide range of applications throughout the arts and sciences. A wonderful application that merges art and science is the generation of artificial plants. Botanist Aristid Lindermeyer started working on artificial plants in the 1960s and from his work came a specialized set of grammars called L-systems. L-systems are used to model the growth of plants. Algebraic equations can model the rate of growth of single cells, tissues and stems, but have a hard time with branching structures; as you can imagine from our parse trees, formal grammars can easily model the production of branches.

An L-system is very similar to a formal grammar, with one major difference; the interpretation of the resulting language. Instead of treating the language as mathematical formulas or English sentences, we will interpret each string as a set of instructions for a drawing a figure. We will use Turtle graphics as we did in Chapter 3 to draw fractal trees, but this time L-systems will substitute for recursive C calls.

Turtle graphics using command strings

To review, Turtle graphics uses an imaginary turtle that draws on in an (x,y) coordinate system according to a few short rules. The turtle has a state (x,y,A), where it is at position (x,y) and pointed a direction defined by the angle A.

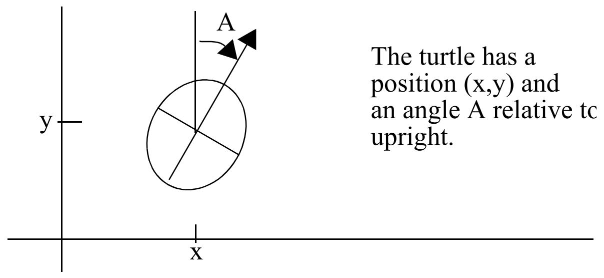

A
y
x
The turtle has a position (x,y) and an angle A relative tc upright.

Figure 9.
Definition of the Graphics Turtle

For our L-system application we will reduce our turtle to three commands, each expressed as a single symbol: F, + and -. F means more ahead a distance d; + means rotate delta degrees left; and - means rotate delta degrees right. The values of d and delta are fixed in advanced of each figure; all the Fs during one session means the same distance d. After a command the turtle updates its state (x,y,A).

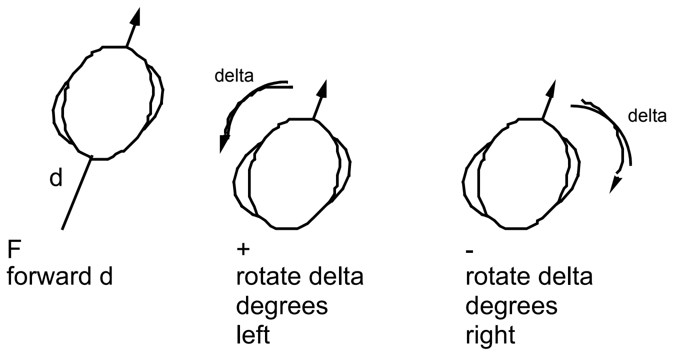


Figure 10.
Definition of Turtle Graphics Commands

We can now command our turtle by giving it a string composed of Fs, +s and -s. To draw a box, we set d = 1 inch and delta = 90 degrees. We start with the turtle facing up at the origin of our coordinate system, so the initial state is (0,0,90). The string for drawing a box would be F+F+F+F, an F for each of the four sides. The figure below shows how the turtle draws F+F to start the box.

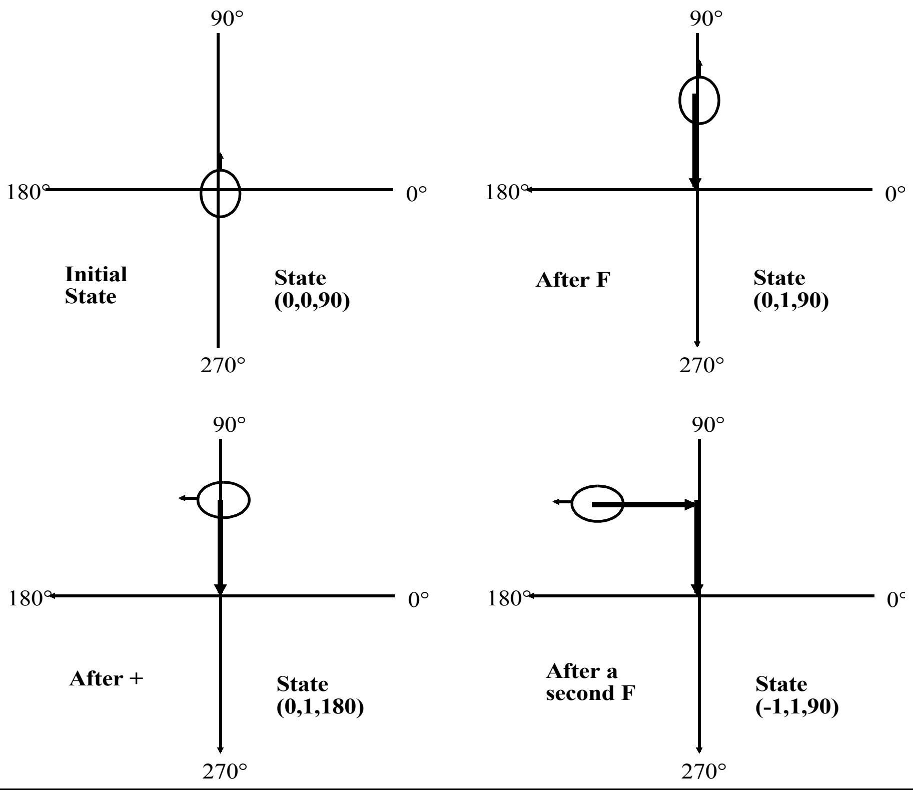


Figure 11.
Example of Turtle Graphic Commands

Drawing a branch

You can draw all types of figures with the simple commands F, + and -, but they do force the turtle to draw in one continuous line as if the turtle had a pen that could not leave the paper. One way to solve this is to give the turtle a little bit of memory, so the turtle can draw one part of a figure, remember where it is, go someplace else and draw another part of the figure, and then return to where it was. To go back to where it was earlier, the turtle must remember its previous state with location and direction. We used recursion in Chapter 3 to remember where the turtle was, but this time we will not.

This time we will give our turtle a *stack* for memory. A stack is a special kind of memory that can store as many previous states as we would like, but that only allows the turtle to retrieve the most recently stored state. Imagine that to remember a state, our little turtle writes down (x,y,A) on a sheet of paper and drops in the sheet on top of a pile. When the turtle wants to retrieve a location, it must pick up a sheet and read the state. But, a stack restricts the turtle to the most recent sheet. The turtle cannot leaf through the pile looking for a particular place, it must take the one on top. When it reads a state, it throws it away.

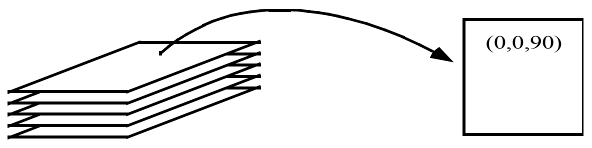


Figure 12.
Example of Turtle Graphic Commands

To command the turtle to remember where it is, we will use two new symbols. An opening bracket, `[`, tells the turtle to place its current state on the stack. A closing bracket, `]`, tells the turtle to take the current state off the top of the stack and move to that location. Here is a example of how to draw a branch with the commands **`F[-F]F`**; d is set to 1/2 inch and delta to 45°.

.

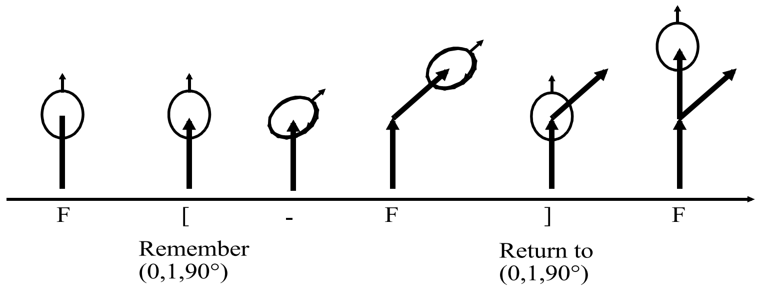


Figure 13.
Example of Turtle Graphic Commands

Here is a example of how to draw a branch with the commands **`F[F[+F]F]-F`** with the same values for d and delta. In this case the stack has to contain two states to handle two branches.

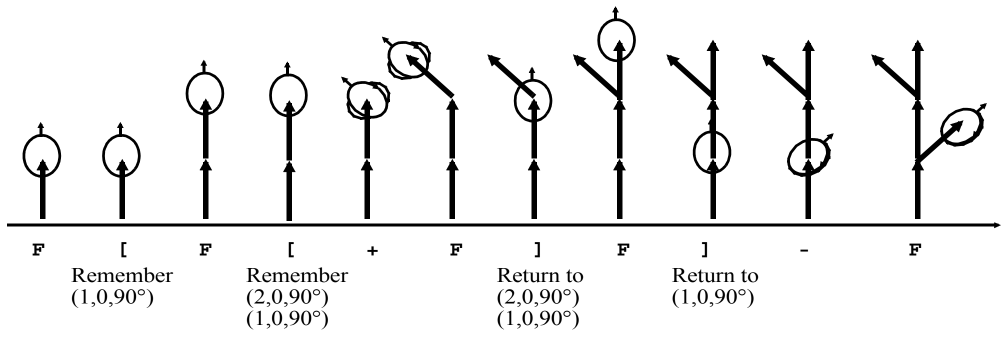


Figure 14.
Example of Turtle Graphic Commands

An L-System using Turtle graphics

Our L-systems will use this modified Turtle graphics to create strings of commands and then draw them. The L-systems use a BNF to generate a string, and then Turtle graphics to interpret the string; Turtle graphics are the semantics of the system. Here is an L-system for a symmetric bush. The system has only one syntax name and one rule.

Formal Grammar: **L-system for a symmetric bush**

Alphabet: **`F + - [ ]`**
Start character: **`F`**
Parameters: Distance = 1, Delta = 45°

```
Rule      F ->
              F[+F]F[-F]F
```

To use this grammar, you generate a derivation of n steps; at each step, you replace all possible Fs simultaneously. So if n=2, we'd have:

```
F   ->   F[+F]F[-F]F
    ->   F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-
         F]F]F[+F]F[-F]F
```

As you can see, the strings get very long very soon, hence the need for a computer to handle the details. For n=3 the string has 301 characters. Once you have a string you send it to a turtle for interpretation, and it draws the bush. Here are three generations of this L-system:

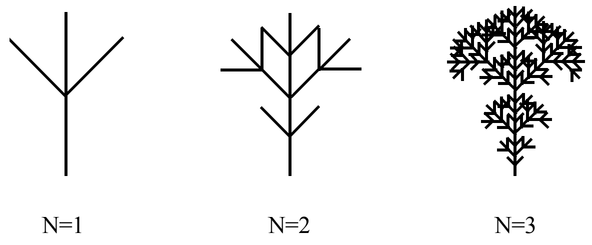
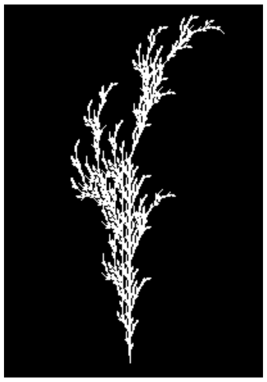
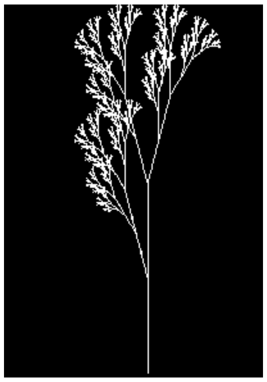


Figure 15.
Example of Turtle Graphic Commands

Other examples of L-Systems

The examples in the next figure appear in **The Algorithmic Beauty of Plants** by Przemyslaw Prusinkiewicz and Aristid Lindermayer. The upper two and lower left are plants, the lower right an abstract curve that could be a coastline. The next figure also comes from this book, but it uses a more complex form of Turtle graphics to draw a model of the bush in a full three dimensions before the picture is created. Another example is included on color plate XX.

n = 5, delta = 25.7°
Rule: F -> F[+F]F[-F]
	n = 4, delta = 22.5°
	Rules:x=f[+x]f[-x]+x
	f=ff

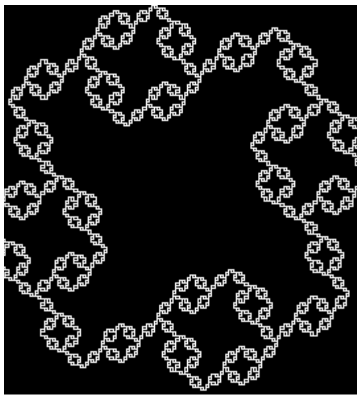
	
n = 4, delta = 90° Rule: f=ff-f-f-f-f-f+f	

Figure 16.
Examples

Figure 17.
L-System plants

L-Systems can be used to draw beautiful pictures of plants, but this is not their only use. They can model the branching structure of plants and animal organs, or abstract mathematical objects, using brief sets of BNF rules. L-Systems demonstrate the power of formal grammars to capture the complexity of natural objects.

Exercises for 8.1.4

1. Draw the figures that result from these turtle graphics strings. Use a reasonable value for d. Assume the turtle starts off pointing up, with A = 90°.
   a. With delta = 90°. F - F + F + F + F
   b. With delta = 45°. F - F - F - F - F - F - F - F
   c. With delta = 90°. F [- F [- F] + F]
   d. With delta = 22.5°. F F - [- F + F + F] + [+ F - F - F]
   This is one of the bushes in the figure above. Do your best with 22.5°; this is half of 45°.
2. Give turtle graphics strings, with an appropriate delta, to draw these figures:
   a. The letter Z
   b. The letter T
   c. The letter Y
3. Out of the 26 letters in the English alphabet, give five that can be drawn without using the stack and five that require it.
4. Here is an L-system for a simple bush. Use the grammar to generate the first and second generations (n=2) strings, and draw the resulting two figures. Let delta = 45°.

```
F -> F[-F]FF[+F]
```