Forward

A User Guide to SNAIL LOGO

Contents

2.2 Introducing SNAIL LOGO

The best known feature of LOGO is its so-called "Turtle Graphics". This idea is based upon a small mechanical turtle, with a pen underneath, which crawls along on a large sheet of paper, drawing lines as it goes. Via a computer, the Turtle can be commanded to perform movements like FORWARD, and TURN RIGHT. Its command language is simple but has surprising power. The direct relationship of the language to the patterns drawn by the turtle also provides an illuminating visual illustration of programming structures and sequences, and allows the programmer to observe the operation of his program. Because it is so easy to use, the programmer may well be a child, who can use the system to explore his or her own ideas. The sense of creative achievement which can come from even a simple program is a powerful motivating force.

SNAIL LOGO has a language and operating system which provides an advanced version of "Turtle Graphics". Instead of an actual mechanical turtle, it uses a graphical snail, which can be made to move around the computer display screen. In the usual manner of snails it leaves a track behind it, so it can draw shapes and patterns.

SNAIL LOGO can show many of the principles used in the design of complicated things – like computer programs. Some of these principles have particular names, like Hierarchy and Replication, but it does not matter whether these names are meaningful at the start. The actual use of SNAIL LOGO will make them clear.

2.3. What SNAIL LOGO explores

In this introduction, some points are illustrated by reference to a motor-car rather than to a computer program. (Many design principles are common between the two!) A young child however can learn the principles from a simple "clean sheet" basis, without being concerned about comparisons.

Computer languages are sets of words for instructing computers to carry out specific activities. There are dozens of computer languages, of many different types. One way of classifying them is by "level". A low level language (like machine code assembler) is able to instruct the computer to switch transistors on and off to represent '1's and '0's. The programmer has to decide how the '1's and '0's are to represent numbers and facts however, which is an intricate task.

A higher level language (like Basic) has a different sort of vocabulary and allows large chunks of computer operations to be called for without troubling about the detailed processes necessary to carry out each chunk.

Differences between languages can be seen in terms of different ways of describing a motor-car. To describe to someone exactly how to build a car, each nut, bolt and component would have to be specified in an exact sequence, and a low level language would be used. In principle it would not be necessary to use the word "engine". That component would be produced by following the instructions for putting its parts together. Such a description however, would be a very poor way to explain to a visitor from another galaxy what a car was, and how it worked. Some much higher level description in terms of large units like engines and doors would be better, and ignoring what engines and doors were made from would not matter.

SNAIL LOGO language is very high level. It has simple and powerful commands like 'FORWARD' and 'REPEAT', which instruct the Snail to move and to leave a line of track behind it. Also, like LOGO, there is no need to be concerned with how this works inside the computer.

There is a reason for SNAIL LOGO having a very high level language. By making it simple to specify the basic operations, it becomes easy to explore the ways in which complicated processes and structures can be built up by combining simple ones in various ways.

Returning to the car illustration, words for parts of a car like seat, engine and carburettor are not only names for working aspects of the car, but refer to parts actually designed and made separately. A car is in fact a group of sub-units which work at different tasks but cooperate to provide an overall function. A computer program is similarly made from a structure of different components which can each contain further components. Such an organisation is called a *hierarchy,* and it will also have a *sequential* organisation. The components of a program can be called *procedures*. Some components are used to assist or service several other components, like the electrical system of a car. Some components work by *repetition,* like a car engine. Sometimes components are *replicated* like the wheels, headlights, and front seats of a car. Other components perform a task which is standard in principle but varies in some way, like a car gearbox, which always connects some gears but can have a number of different ratios. Such a variable within a component is refered to as a *parameter*. Also, systems like cars or computer programs can be in different *states,* like braking or accelerating, and running or crashed, and these states can be defined by many separate *state variables*. Another important process, which does not have a mechanical analogy but which is useful in computer programs, is known as *recursion*.

SNAIL LOGO allows the various aspects of creating and using these techniques for definition and expression to be explored. All of them are reflected in the SNAIL LOGO language capabilities, and can be used in designing graphics programs. They can be observed in operation because the "Snail" makes its tracks at a speed which allows the graphics program to be followed as it runs.

A last general point. Learning most computer languages is a rough equivalent of learning how to put together nuts and bolts, or bricks and mortar. Learning to draw graphics with SNAIL LOGO is more the equivalent of learning how to design motor-cars, or houses (and computer programs of course!).

3. STARTING TO USE SNAIL LOGO

3.1 Introduction

This section leads you into the use of SNAIL LOGO by describing practical examples. Because this is just to start you off, it only gives a small sample of the potential of SNAIL LOGO. After studying the rest of the guide you will be able to expand and enhance the examples and explore and develop your own ideas.

In case of problems, refer to the other sections of the Guide – everything is explained.

Before starting the examples there are some important terms to note:

Instruction, Program and Procedure.

There is no need to remember all the details at once – read the descriptions below again when you need to.

Instruction: A SNAIL LOGO Instruction is made up of a Command name, sometimes with values given as a number or a variable name following the command. Instructions either cause some action to be taken by the Snail, or affect the operation of other Instructions. All the Instructions are shown in Chapter 5, with explanations of their operation. General principles are as follows:

(a) The Snail always has a position, and a direction. It changes its position by the number of steps given in the FORWARD and BACKWARD Instructions, and its direction by the number of degrees given in RIGHT and LEFT Instructions. (CENTRE, POSITION, NORTH and RNORTH also cause changes).

(b) The Snail itself can be visible or invisible, according to how SNAIL and NSNAIL Instructions are used.

(c) The track of the Snail can be visible or invisible, according to whether its pen is DOWN or UP. If its track is invisible, single points of track at its current position can be shown if a OUTPUT Instruction is used. The COLOUR Instruction defines the colour of the track.

(d) REPEAT and PROCEDURE Instructions affect other Instructions, and are described below.

Program: A program is a list of SNAIL LOGO Instructions which are obeyed one after the other when the program is run. The REPEAT and RFINISH Instructions cause all the Instructions in between them to be repeated the number of times stated in the REPEAT Instruction. A program must always finish with an END Instruction. If variables are used they can be manipulated by SET, INCREASE, DECREASE and MAKE Instructions, and inspected by the SHOW Instruction. A program or a procedure can be made to end only under certain conditions by the IFEND Instruction.

Procedure: A procedure is a list of Instructions just like a program. However a procedure can not be run directly. Only programs can be run. Procedures are used by programs or other procedures. Procedures each have a name (like SQUARE or CIRCLE) which you can choose, usually to indicate what the group of Instructions which they contain will do. Thus, programs can use the whole group of Instructions in a procedure just by using the procedure name. For example: PROCEDURE SQUARE.

3.2 The Examples

When you follow the directions given for the examples in sections 3.3 to 3.8 below, your computer will show the results on its display. Think about the relationship between the SNAIL LOGO Instructions used and the patterns produced.

Before starting section 3.3 have a look at the list of SNAIL LOGO Instructions in section 5.1, and refer to them as necessary to understand what each Instruction is doing.

SNAIL LOGO is operated from its "Menu" display. The Menu is a list of facilities displayed on the screen, in three groups. By entering the single letter code (except for one case of three letters), the facility is obtained for use. After use, the facility returns to the Menu so that the next can be selected. Facilities can only be obtained from the Menu when it is actually displayed.

The ENTER key is usually all that is required to step on from one stage of a facility to the next, and eventually return to the Menu. However, if entering a program, editing a program, or entering a procedure, that operation must be finished first by entering END, or cancelled, before the ENTER key can be used to return to the Menu. Read section 4.3(1) NOW, to make sure that you understand this, and note the use of the Z and U keys. Often the ENTER key must be used several times to step through a facility to get back to the Menu.

An important point:– you may find that the various checks and error messages which arise are annoying! However you will soon get used to the simple rules, and the checks are very important, particularly for children. As far as is practicable they make sure that your SNAIL LOGO program is sensible before you RUN it, so there won't be silly or confusing results.

Until you alter it by a program, the Snail always starts in the centre of the screen, pointing "North", with its pen down, and invisible.

Press any keys you like. You will not do any harm to your computer or SNAIL LOGO, though if you try hard enough, BREAK might stop SNAIL LOGO running. If you do, just reload from tape.

Early examples describe facility use in detail. Later examples refer to the use of facilities without repeating all the details. Refer again to earlier examples if necessary.

Finally, if you do not know what to do at any stage, always try pressing the ENTER key.

3.3 First Steps

- Read all of each paragraph before trying it.
- All the keys mentioned must be "entered", that is, the particular key must be pressed, then the ENTER key. Nothing will happen until the ENTER key is pressed.
- If you do not have a printer, ignore references to entering C. Use just the ENTER key instead.

(1) Load the SNAIL LOGO system tape as described in section 4.2. When loading is complete the Menu will appear. It is titled SNAIL LOGO followed by some dots and a Snail symbol.

(2) CAPS LOCK will be set automatically as only capital letters are used in SNAIL LOGO.

(3) On the Menu, enter R to RUN a demonstration program included on the tape. It is not a typical SNAIL LOGO program but illustrates a number of techniques, including a particular use of the Snail symbol.

(4) Enter C if you want a printed copy, or just ENTER to return to the Menu if not.

(5) Enter L on the Menu to list and inspect the Flower program.

(6) Press ENTER to obtain a (P)ROC? message. Ignore this.

(7) Use ENTER again to obtain a (C)OPY? message. Enter C and the program listing will be copied to the printer, then a return made to the Menu.

The listing has an abbreviated form of commands using only the first two letters of each command name. Check the full versions of the commands using section 5.1, but there is no need fully to understand the program at this stage.

(8) Enter K on the Menu. Next enter the procedure name STEM. Have a look at the procedure listing. To print a copy, respond to the (C)OPY? message by entering C. If you do not require one, return directly to the Menu with ENTER.

(9) Use the method described in (8) to have a look at the procedures LEAF and BLOOM. Use ENTER to return to the menu after each one.

3.4 Next Steps

Now to enter some Instructions which make up simple SNAIL LOGO programs, and run them. To enter an Instruction, the whole Instruction such as FORWARD 29 is typed, and appears at the bottom of the screen. The Delete key can be used in the usual way at that stage. When the Instruction is complete the ENTER key is pressed.

(1) Enter EPR on the menu. (You have now overwritten the Flower program). The system is now ready to accept program Instructions.

(2) Enter 123ABC. The result is an error report. (Section 5.4 describes the various types.) ENTER to cancel it. (EN on the screen means ENTER).

(3) Enter FORWARD 6 (Note the single space). Enter END. That is an accepted program. ENTER twice to return to the Menu.

(4) Enter R on the Menu to run the program, and watch the display. Check the 6 forward steps taken by the Snail. ENTER to return to the Menu.

(5) Enter EPR on the Menu, and then the following program, (using ENTER after each Instruction).

```
REPEAT 6
BACKWARD 1
RFINISH
END
```

(6) Return to the Menu and run the program with R. Notice the effect of REPEAT and BACKWARD.

(7) Enter L on the Menu to list the program. ENTER to get the (P)ROC? message, then enter P to convert the program to a procedure. A request for a procedure name will be shown. Enter SIXBAC. A request for a procedure number will be shown. Enter 1.

(8) Enter the following program, (EPR), which uses the procedure SIXBAC, and run it.

```
REPEAT 2
PROCEDURE SIXBAC
RFINISH
END
```

Notice the effect of using the procedure twice (by repeating it).

(9) Enter and run the following programs, given with abbreviated commands. Use chapter 5 to check the command names. Whilst entering these programs investigate the effects of entering U and Z.

(a)
```
SN
RI 90
FO 8
END
```

(b)
```
PO 10,10
RE 4
RI 90
FO 8
RF
END
```

(c)
```
RE 9
FO 15
RI 160
RF
END
```

(10) Enter P on the Menu, and then give the name SQUARE to a procedure. Give it the number 2. Write the procedure Instructions to give a square of 6 steps each side. (Hint – look at example 9(b) above). Enter and run a program to use procedure SQUARE. (Hint – look at (8) above, but there is no need to REPEAT the procedure).

(11) Use N on the Menu and see what happens.

(12) Try entering and running this program:

```
RE 3
PR SQUARE
RI 120
RF
END
```

(13) Enter E on the Menu for Edit. Enter line number 3. Enter R for Replace. Enter the Instruction RI 90 (to replace the Instruction RI 120), then END. When the changed program has been listed, enter Z to finish the Edit. Run the revised program.

(14) List the program and convert it to a procedure named BLOCKS, number 3. Enter and run the following program:

```
PO 20,20
PR BLOCKS
PO 40, 20
PR BLOCKS
END
```

Note how a hierarchy of procedures may be built up, each using other procedures. How would you make the two Blocks appear the same way around?

3.5 Variables

The next set of examples introduces the use of variables. These add considerably to the power of the SNAIL LOGO language.

A variable is a numeric value which has a name. It is referred to by its name, so the number which it represents can be any value and can change, which is why it is called a variable.

SNAIL LOGO has eight different variables, named A,B,C,D,E,F,G,H. They can all have any positive or negative numeric value. Only positive whole number values can be entered directly, but negative or fractional values can be obtained by performing arithmetic on the variables.

The variables do not all behave in the same way. There are two different types, "Global" variables and "Local" variables.

Global Variables: E,F,G,H

A Global variable is available to, and can be used by, any program or procedure in the SNAIL LOGO system. If some procedure called APPLE sets E to a value of 17, procedure ORANGE can read and use and change that value. There is only one E, one F, one G, and one H.

Local Variables: A,B,C,D

There are many copies of each local variable. Every procedure and the program has its own separate copy. If the program sets A=3, procedure APPLE cannot see or use that value of 3. APPLE might set its A=24 and this would not disturb the value of 3 in the program's copy of A. Procedure ORANGE can use yet another different value and all the values of A would exist separately.

There is one way in which a program or procedure can set the values of A,B or C (not D) for another procedure. That is by using the PROCEDURE instruction. PR APPLE, 9 not only calls the procedure APPLE, but sets the value of procedure APPLE'S copy of A to 9. The PROCEDURE instruction does not affect the value of the copy of A for the program or procedure in which it occurs. The use of this facility is described below under the Procedures and Parameters heading in section 3.6.

Variable D

Variable D is basically an ordinary local variable and can be used as such. However it is used by the SNAIL LOGO system to hold the direction in which the Snail is pointing on entry to programs and procedures. This allows the RNORTH (Relative North) command to operate, by enabling the "North" direction to be rotated as a procedure is rotated. If D is altered by Instructions, RNORTH will not work correctly, though this effect can be useful in advanced programs.

Variables can be used in many Instructions, such as FORWARD A and REPEAT F. However they have to be given values before being used. Initially SNAIL LOGO sets all variables to 0. Other values can be set by a PROCEDURE Instruction as described above, and by several other types of Instruction.

SET	directly gives a value	e.g. SET C,19
INCREASE	adds a number to the present value	e.g. INC F,3
DECREASE	subtracts a number from the present value	e.g. DEC G, 1
MAKE	allows arithmetic to be carried out	e.g. MAKE A = B + C MAKE C = F / G MAKE D = H * A MAKE E = D – A

To see the value of a variable on the display screen as the program runs, use the SHOW Instruction e.g. SHOW C.

Note that for multiplication using the MAKE Instruction, the star symbol must be used, as for Basic.

(1) Enter and run the following programs:

(a)
```
SET A, 0
RE 6
INC A,20
FO 6
CE
SH A
RI A
RF
END
```

(b)
```
SET A,4
SET B,5
MA G=A+B
MA D=A/B
SH G
SH D
END
```

(2) Input the following procedure TAR (give it the number 5) by entering P on the Menu, then enter the program to use it, and run it.

Program
```
SE H,5
RE 4
FO 5
PR TAR
CE
RI 90
RF
END
```

Procedure TAR
```
RI 90
DE H,1
FO H
RN
END
```

This program illustrates both the use of the RN Instruction (try replacing it with NORTH), and the global nature of the H variable.

3.6 Procedures and Parameters

These examples illustrate the use of parameters in procedures. Look at the general form of PROCEDURE Instructions in section 5.1 before starting.

The variables A,B and C used in a procedure can be given values by the PROCEDURE Instruction which calls that procedure. For example:

PROCEDURE APPLE,3,10,90

The first number given after the procedure name sets the value of A in procedure APPLE, the second sets B and the third sets C. This technique is referred to as passing parameters to a procedure. Thus a program may call the same procedure a number of times and obtain different results by passing different parameters. For example, three different sized squares could be drawn with the three Instructions:

PROC SQUARE,3 PROC SQUARE,5 PROC SQUARE,10

Note that if a PROCEDURE Instruction is included in a REPEAT loop, the parameters are passed only on the first call. Thus the parameters are not reset on each loop cycle.

(1) Use P on the Menu and enter the following procedure named CIRCLE (Number 4):

```
RE 8
FO A
RI 45
RF
END
```

Enter and run the following program. Notice the parameter value of 2 given to the procedure, which is then used as the value for A in the procedure.

```
RI 20
RE 4
PR CIRCLE, 2
LE 90
RF
END
```

(2) Use the Edit "Replace" facility to change line 3 of the program to PR CIRCLE, 4 and run it again. Then again use Edit to "Insert" the Instruction PR CIRCLE, 2 as well as, and next to, PR CIRCLE, 4 (Use either line number 3 or line number 4). Run this program.

(3) Write the following procedure, POLY (decide on your own numbers from now on), and a program to run it with different values of A.

Procedure POLY

```
MA B=E/A
RE A
FO 6
LE B
RF
END
```

Program

```
SET E, 360
PR POLY, 3
PR POLY, 5
PR POLY, 7
END
```

Using the relationship that A times B = 360 gives this particular kind of result. Try other relationships.

If you have two minutes to spare, this program is worth waiting for. It uses POLY. The same effect could be obtained, and would run faster, by writing it as a single program.

```
SET E,360
RE 16
PR POLY,8
RI 22.5
RF
END
```

(4) Try this:

```
Procedure SPIRAL
FO C
IN C,1
RI 90
END
```

```
Program
RE 50
PR SPIRAL,1,1,1
RF
END
```

Note that the values for A and B are dummies, necessary to set C. Stop this before completion if you wish, by pressing Z.

(5) Write a procedure TRISPI, and a program to use it, that will give a triangular spiral effect.

(6) Try this:

```
Procedure SPITRI
FO A
IN A,1
RI 120
END
```

```
Program
RE 50
PR SPITRI,1
RI 5
RF
END
```

(7) Try this:

```
Procedure GALAXY
FO A
IN A,1
RI 149
END
```

```
Program
UP
RE 50
OU
PR GALAXY, 1
RF
END
```

Experiment with different angles in the GALAXY procedure.

(8) Finally note that the simple procedure and program below can be used for many shapes, simply by using different parameter values. Try the example given:

```
Procedure FLEXI
RE A
FO B
RI C
RF
END
```

```
Program
PR FLEXI,3,4,720
END
```

3.7 Recursive Procedures

As has been shown, hierarchic procedures can be written so that a program may call a procedure HOUSE, which may call a procedure DOOR, which may call a procedure KNOB. However a procedure may also call itself; this process is known as recursion.

(1) Try this:

```
Procedure FRED          Program

FO 1                    PR FRED
PR FRED                 END
END
```

(2) Variables may be used, and changed, for example:

```
Procedure JOE           Program

FO A                    PR JOE, 1
IN A,1                  END
PR JOE, 1
END
```

If Bounce is set and Z is not used, both of the above examples will run until the recursion stack is exhausted, when an error message END OF RECURSION STACK will be given. To avoid this, the IFEND Instruction is useful.

(3)
```
Procedure JIM           Program

SET B, 5                PR JIM, 1
FO A                    END
IN A, 1
IFEND A>B
PR JIM, 1
END
```

The conditions for an IFEND Instruction may use local or global variables. The latter may be set by other procedures or by program. (How could example 3 above be speeded up slightly by using a global variable?) Note that IFEND may be used in any situation, not just with recursive procedures. Use within a Repeat loop will stop execution at that point however.

Careful examination of the running of the examples (2) and (3) above will show that two single steps are taken before A increments to 2,3 etc. This illustrates an important point. When a procedure calls itself recursively, the same copies of its local variables are used, so that changes to variables made in the procedure are effective the next time the procedure calls itself. However, this set of variables is not the same set as allocated by the program when it first calls the procedure. Thus on the first call, the program may set the variables and on the recursive call within the procedure the values of the variables may again be defined, not necessarily to the same values. It is only on the first recursive call that the initial value settings will be applied however, else changes made to variables during the procedure would

be cancelled on each recursive call. It should now be possible to see why two single steps are first taken by the above examples. Try experimenting with different values and plenty of SHOW instructions to see the effects. Use an initial value of 0 in the program call to avoid an initial movement, as in the next example.

(4) Try this:

```
Procedure REC
RE 8
FO A
RI 45
RF
IN A,1
IF A>H
PR REC,1
END
```

```
Program
SE H,5
PR REC,0
END
```

(5) Try this:

```
Procedure SQR
RE 4
FO A
RI 90
RF
IN A,1
RI 25
PR SQR,1
END
```

```
Program
PR SQR, 1
END
```

3.8 Calculations

Simple calculations related to shapes may be made as follows:

(1) Perimeter Length

```
Procedure MOV
MA H=H+A
FO A
SH H
RI B
END
```

```
Program
SE H,O
RE 2
PR MOV,6,90
PR MOV,4,90
RF
END
```

(2) Area

Areas can be calculated in a generally similar manner to perimeters but the area must be sectioned into rectangles and triangles. This can be shown effectively by first drawing an outline, then sectioning it and calculating the area of each section as it is drawn in, with a second variable to sum the running total. Colour is effective here. A variety of detailed techniques may be used, and the exercise is left as a project for the user.

4. THE SNAIL LOGO OPERATING SYSTEM

4.1 Introduction

This chapter of the Guide does not directly deal with the SNAIL LOGO language or its use. Rather it covers all the aspects of operating the SNAIL LOGO system on your computer. If you are not already familiar with the usual features of an operating system, it does not matter. They are all provided in a simple to use form and allow the SNAIL LOGO language to be used in a flexible way:

- To enter a sequence of language Instructions as a program.
- To warn of incorrect or impossible Instructions.
- To make changes, deletions or additions to a program.
- To enter a sequence of Instructions as a procedure, or to convert a program to a procedure, or to cancel a procedure.
- To list a program or a procedure on the screen, and print it, and to list the names of presently defined procedures.
- To run a program, and draw the resulting Snail track on the screen, then print the result.

4.2 Loading the SNAIL LOGO system

(1) Read your computer handbook to check the general points regarding loading programs from tape.

(2) Insert the tape in the tape recorder, and LOAD "".

(3) After a few seconds the title display should appear. If it does not, adjust the volume setting on the tape recorder and try again. To complete loading then takes a further 1½ minutes. When loading is completed, the main menu is shown.

4.3 Operating System Facilities

(1) General Aspects

The Menu display is introduced in section 3.2. All operating system facilities are accessed from the Menu, and return to the Menu after use. Access them by entering the code letter or letters shown against each facility on the Menu.

In general, use the ENTER key to progress through a facility and return to the Menu. The one exception to this arises after Instruction entry has been started. This may be for a program (after EPR), for a procedure (after P), or for insertion of Instructions during an Edit. Once Instruction entry has been started it must either be finished with an END Instruction, or cancelled by entering Z, before it is possible to return to the Menu. This applies even if no Instructions have actually been entered.

Enter Z to cancel an Instruction entry process. Enter U to delete just the last Instruction entered, though U may be used repeatedly to delete a number of Instructions.

Each facility is described below. The easiest way to become familiar with them is to use them! General guidance and error messages are provided throughout.

All SNAIL LOGO inputs need CAPITAL letters, therefore make sure that the CAPS LOCK remains set.

Have a glance at the reference list of language commands in chapter 5 before studying the rest of this section if you have not already done so.

(2) Enter Program (EPR)

This facility allows SNAIL LOGO Instructions to be entered in sequence to form a program, entering each one with the ENTER key. Because entering a new program cancels the previous one, a 3 letter code is used to minimise the chance of accidents.

Line numbering is automatic. The simple organisation of command words and numbers (the Instruction syntax) is shown in chapter 5. One space is required between commands and a number or name, and commas are used to separate variables and numbers. If Instructions do not have a correct syntax, or can not be understood, an error report is given. A list of these reports appears in section 5.4. Following an error report use ENTER to cancel that Instruction. Only the first two letters of each command name are checked, therefore the other letters can be omitted and an abbreviated form used, which is usually preferred after a little use. (This also allows commands like FOSSIL 19 and CLOUD to be input and accepted as FORWARD 19 and CLEAR however!)

The last Instruction in every program (even for a "do nothing" program with no other Instructions) must be END.

Instructions which are within a Repeat loop are indented in the listing, making the repeated sections easy to see. Nested (that is, inside each other) repeats are further indented.

The number of REPEAT and RFINISH Instructions in a program must be equal. If they are not, a message saying RF NOT EQUAL RE gives a warning when you try to END, and does not permit the END Instruction. Press ENTER to cancel the message, then add RE or RF Instructions as necessary. The indentation makes this very easy to check. Almost always, one more RF is needed. Use U and Z as necessary.

After END is used, an ACCEPTED message is given, which means what it says, and at that stage two successive messages (P)ROC? and (C)OPY? indicate that the program can, if you wish, be converted to a procedure by entering P (followed by a procedure name and number), and/or copied to a printer with C.

(3) List Program (L)

This gives a listing of the current program, and after an ENTER, the standard options are given to convert the program to a procedure, or copy it to a printer.

(4) Edit Program (E)

This facility allows changes to be made to a program by replacing, deleting or inserting Instructions. Start by entering a line number. Deletions and replacements are of the specified line. Insertions are made before the specified line, so use a number one more than that of the last line to add extra Instructions at the end of the program. If a "silly" line number is given, there is a return to the Menu (via (C)OPY?) if the number was less than 999. (Rubbish inputs are ignored, as in other aspects of SNAIL LOGO).

After specifying a line number the option to Replace, Insert or Delete is given. Enter R,I or D. For either R or I, a heading of "Inserted Instructions" appears and Instructions can be entered in the usual way. Note that the Instructions entered are line numbered from I at this stage.

Finish a set of "Inserted Instructions" with END to return to the Edit line number selection stage. Note that END is necessary to return, even if no Instructions have actually been inserted. A second line number can then selected for a further edit process. When all changes are complete, return to the Menu with Z. The Menu return is via the (C)OPY? option.

Note that when inserting or replacing Instructions, any number of them can be put in together. Note also that when using Edit, single RE or RF Instructions can be added or deleted (unlike when Entering a program, when only pairs of an RE and an RF are accepted). Warnings are given but "silly" sequences of RE and RF can therefore be generated by the Editor. Trying to run these will lead to an error message.

A procedure named in a program PROCEDURE Instruction can be deleted by having its number used by another procedure definition. If a program including a PR Instruction with a procedure name which no longer exists is edited, an error report will be given, and the PR Instruction omitted from the edited version of the program.

(5) Enter Procedure (P)

Procedures can be generated directly with this facility, First a name for the procedure must be specified, then a procedure number, from 1 to 9 . Any other number will cause a return to the name entry stage. If either name or number is omitted there is a return to the Menu. Different numbers must be used when more than one procedure is needed, as use of a number causes any other procedure with that number to be overwritten. If two procedures have the same name, that with the lower number is used when that name is called by a program.

Procedures must be defined i.e. named, numbered and entered (the Instructions can be simply END for a "do nothing" procedure) before trying to refer to that procedure by name in a PR Instruction.

SNAIL	– Causes a Snail symbol to be displayed at the end of each track produced by the following instructions.
NSNAIL	– Cancels the effect of SNAIL.
DOWN	– Causes Snail tracks (and symbol) generated after this instruction to be visible.
UP	– Causes Snail tracks (and symbol) generated after this instruction to be invisible.
OUTPUT	– Causes an element of Snail track to be printed at the current position.
CENTRE	– Moves the current position to the screen centre.
NORTH	– Sets the current direction towards the top of the screen.
RNORTH	– Sets the current direction to the initial direction when the program or procedure started.
CLEAR	– Clears the screen.

PROCEDURE name

PROCEDURE name,n,n,n (1,2 or 3 n's may be specified)

PROCEDURE Instructions action the Instruction sequence which has previously been defined by the name. If desired one, two or three parameter values may be passed to the procedure, by including them as shown after the name, separated by commas. Within the procedure the parameters are identified as A, B and C. The parameter values in the PROCEDURE instruction are always assigned in the sequence A, B, C.

END – Must be used as the last Instruction in programs and procedures.

Instruction	**Range for n and N**
FORWARD, BACKWARD REPEAT, SET INCREASE, DECREASE	Whole number 0 to 999
RIGHT, LEFT PROCEDURE	Whole number 0 to 999 and the value of 22.5
POSITION	1 to 62 for horizontal 1 to 42 for vertical
COLOUR	0 to 6

5.2 Notes on Instruction Use

(1) All Instructions may be abbreviated down to a minimum of the first two letters, except **END**.

(2) REPEAT Instructions must be followed by RFINISH at some point. Repeat loops may be nested within each other, and multiple loops may be used, both within programs and procedures.

(3) Initial values of Variables are zero.

(4) A Snail symbol can be removed by rewriting it at the same location. It can be displayed at the current location by the sequence SN FO 0. (Try SN FO 0 FO 0).

(5) When A, B or C are used in procedures, values can be specified in the PR Instruction when the procedure is called. A dummy value must be given for A and B, if C alone is to be specified, and for A if B alone is needed, to maintain the sequence A,B,C.

(6) Before any Instructions are given in a program, initial conditions are CENTRE, NORTH, DOWN, NSNAIL, COLOUR 0.

(7) For colour number definitions see computer manual.

(8) RE0 and RE1 both give a single pass through instructions.

5.3 Operating Limits

(1) Running the examples and some of your own will show how short the typical SNAIL LOGO program is. The following limits should be noted in that light:

Screen size	:	62 steps by 42 steps
Max Procedure Length	:	15 instructions or 70 chars
Max Program Length	:	45 instructions or 200 chars
Max No. of Procs	:	9
Max Length of Proc Name	:	7 chars (alpha only)

Max No. of chars in parameter values is 12 including decimal points.

Max No. of proc calls in a program or a procedure is 10

Max nested depth of Repeat loops (program plus procs) is 15.

These maxima are rarely encountered. The abbreviated form of Instructions is the normal one, so the 70/200 characters limit is ample. The normal program is under 20 Instructions, so can fit one screen. A paging mechanism is provided for exceptional cases.

(2) Exceptionally long Instructions within nested repeat loops, (which are indented for each loop), can over flow on to the next line of the screen. This does not cause any problem either for the program listing or the editor, unless there is more than one, and the screen is full, and the 200 character limit has not been hit. In that case screen overflow can occur and a BASIC "Continue" Instruction be needed, though this is most unlikely to occur.

5.4 Error Reports

Error and warning reports are given under various circumstances. In all cases use ENTER to clear the report. Subsequent actions depend upon the report, and are indicated below.

(1) Instruction Syntax Errors

COMMAND ERR
The command word in the Instruction has not been recognized.

NUM IN COMMD
A number has been found in the Instruction command. Usually caused by omitting a space.

NO SPACE REQ
Some inappropriate space has been found in the Instruction.

NUM/NAME ERR
Some inappropriate characters have been found in the part of the Instruction which should be a number, a parameter value or a procedure name.

NUM NO GOOD
A number outside the range given in section 5.1 has been found.

NUM/NAME REQ
A number, parameter A/B/C or procedure name has not been given.

NUMBER ??
Some character other than a number has been found where a number is expected.

NOT VAR NAME
Some character other than A to H has been found where a variable name is expected.

PROCEDURE ??
The procedure name given is not recognised.

For all Instruction errors the offending Instruction is cancelled by the ENTER used to clear the error message.

(2) Other Errors and Warnings

10 PROCS MAX
At attempt has been made to use 11 PROCEDURE Instructions in a program or a procedure.

REP LOOP ERROR
An invalid REPEAT/RFINISH sequence has been found during a program run, e.g. RF. . . . . .RE, or RF and RE in unequal numbers. The run is aborted and ENTER returns to the menu.

REPEAT LOOP ONLY PART EDITED
A warning given by the Editor, intended to prevent the problem indicated immediately above.

ONLY 15 INSTRS IN PROCEDURES
An attempt has been made to use 15 Instructions other than END in a procedure, or a conversion of a program to a procedure. (There would therefore be no room for the END).

NO SPACE LEFT
Arises if a procedure exceeds 70 characters or a program 200 characters. Unlikely if abbreviated Instructions are used. The program or procedure is cancelled by ENTER.

PROC NAME CANCELLED
Shown after an ENTER to clear the message arising from the two cases immediately above.

PAGE FULL/CONT
Shown when a program exceeds 20 Instructions. Use ENTER, and a new page will be generated, headed by CONT. Line numbers will restart from 1 however, so the Editor can not be used on subsequent pages. A program of up to 45 Instructions (on 3 pages) can be entered and run.

WALL
Shown when a program run has caused the Snail to hit the edge of the drawing area (even if Snail is invisible at the time!). ENTER returns to the Menu.

RF NOT EQUAL RE
See section 4.3(2)

END OF RECURSION STACK
Procedures may call themselves up to a maximum of 50 times only.

REP LOOP IFEND
A Repeat loop has been terminated by an IFEND Instruction. Execution of program can not continue.

DIVIDE BY ZERO
An attempt has been made to divide a variable by a variable having a value of zero.

Appendix 1 - Use of Zeaker Micro-turtle

1 Introduction

Although SNAIL LOGO is complete and self-contained, it can also be used to control the Zeaker micro-turtle, either as an alternative to the Snail track, or in parallel, when both Snail and the micro-turtle will operate. All of the standard SNAIL LOGO instructions may be used with the micro-turtle, though the following notes on their operation applied:

SNAIL - Causes the micro-turtle to operate its horn at the end of each movement.

OUTPUT, CENTRE
NORTH, RNORTH
CLEAR, COLOUR
POSITION

\- Are inapplicable to the micro-turtle and do not have any

2 Control Facilities

The micro-turtle is controlled by a sub-menu selected by the letter T on the main menu. The main menu indicates whether the Snail, the micro-turtle, or both are currently selected by showing the Snail symbol and/or the letter T against the RUN option.

On the sub-menu the following facilities are provided:

a) To select micro-turtle control and turn the Snail display off : T

b) To select both the micro-turtle control and Snail. : X

c) To turn micro-turtle control off and to select the Snail display. : O

d) To adjust the length of a "turtle step", by a number 1-9 : S

e) To calibrate right and left hand turns by setting coefficients in the range 1 to 96. These should be separately adjusted to give accurate right and left 90 degree turns. : K

f) To select the output drive port addres for the micro-turtle. Normally this should be left set to the standard value of 63. : P

Standard initial settings for facilities (d), (e) and (f) are 1, 50 and 63 respectively. Snail/micro-turtle status and all current values are displayed on the sub-menu.