HOW TO STRUCTURE YOUR PROGRAMMING CODE

I remember my first fumble with basic on my ZX Spectrum computer back in the 1980s, plowing through pages of basic commands and example code without any real idea of how I could write programs myself. It was like reading a dictionary where I could learn certain words and their meanings with limited information on how I could construct them into entire sentences to write a document. Every programmer who has dabbled in basic has probably come across the famous "Hello World" routine which consists of a two-line program that prints this phrase unlimited times on the screen.

Your program code needs to be written as step-by-step instructions using the commands that your choice of programming language understands. It means reading your programming manual to learn which commands you need to use for what you want your program to do. In the "Hello World" example you would first need a command that prints "Hello World" onto the screen, and then you would need a second command to print it again multiple times, without writing multiple print statements.

Check out this example. To make things simple I am using old-school basic with line numbers.

```
10 print "Hello World

20 goto 10
```

The best structure for writing any program code is to make it clear and easy to follow. Some programmers put multiple commands on one line which can make the code difficult to follow if you are trying to iron out bugs. Spreading your code over multiple lines makes the program work better and more readable.

Another recommended practice is to separate each part of your program code using REM Statements. REM (short for Remark) allows you to put comments before each section of code to remind you what each part does. This is especially useful if you wish to edit your code at a later date.

```
10 rem Set Up Variables

20 let A=1 : let B=2

30 rem *******

40 rem Print Variables to Screen
```

```
50 rem *******

60 print A,B
```

Anything after the REM command is ignored by the computer. You can use as many REM statements as you want to make bigger gaps in your code for easy reading. Other programming languages allow you to use blank lines or indent the first line of the routine.

Now I will show you how to structure the entire program code. Remember that the computer needs to follow step-by-step instructions so you need to write each instruction in the order you want it to run.

CONSTRUCTION OF CODE

Setup screen resolution and variables: The first section of your program would set the screen resolution and the variables.

Read information into arrays: If you have the information you want to put into an array using the DIM command then you can use a For/Next loop and the READ command. It is best to place the data statements for the array to read from at the end of your program.

Set up the main screen: This is the section where you would use a subroutine (GOSUB Command) to set up the main screen. In a shoot-em-up type game, you would have a routine that draws the sprites and game screen and then returns to the next line of the code it came from.

Main Program Loop: Once the program is up and running the main program loop jumps to various routines using subroutines and then returns to the next line in the loop.

Program Routines: It is a good structure to place all the program routines after the main loop. You would have separate routines that update the screen, check for joystick input, check for collision detection, and so on. After each check, you return to the main loop.

Data Statements: Finally, you can list all the data statements at the end of the program which makes them easier to find and correct if need be.

CONCLUSION

Creating your code with plenty of REM Statements and short lines makes your code look cleaner and easier to follow. There may be a time you want to improve the program or use the routines for other programs.