There are many different factors when it comes to programming languages.  They different not only in their syntax, but expression, statements, procedure to run the code, and even how space is allocated to the program. The differences in the languages can be brought to a simple idea of how that language is controlled. Languages such as C, Ada, and Fortran are all controlled differently given these differences.

A main control that is given to languages is dealing with expressions, conditional statements, and loops. When looking at the category of expressions, we deal with how expressions are evaluated, this can be broken up into three separate categories: infix, postfix and prefix. When we are evaluating prefix and postfix we do not need parenthesis to disambiguate the order of evaluating the expression. This also deals with sequence operators, which determine how things are evaluated, most of the time it is evaluated left to right. Using this some programs contain a functionality called short-circuit evaluation which, will evaluate the Boolean/logical expression up to a point where the entire truth value is known and it will stop. This applies to the idea of loops and logical statements. Most common types of these guard statements are if and case statements. These statements are a staple in the control of programs and they evaluate for Boolean and logical problems. Loops are also an important part in program control, in that they allow for repetitive operations, usually arrays. This allows for faster computation. Loops have a general form that is shown from Dijkstra’s guard if. The statement repeats until all of them are false. In other languages they can different at how they are run. Do while and while loops are similar in function, but do while will run the loop at least one time, while just using a while loop may not run at all. More versions of loops are for-loops where we iterate over a set of values, starting from the beginning to the end. For-loops are in a sense run in its own block, so the variables that it creates cannot be accessed outside of it. For-loops have a big constrain on them in that values of it is counter over cannot be changed in the loop, values that it defines in the loop will be undefined when done, and may other constrains. These critical pieces in programming are what gives the program it's basic controls.

However basic controls of programming languages are not just what the rules the language implements and sets up, but rather a big portion deals with environment and procedures. This deals with topics such as memory and call stacks. A way that programs are controlled are where and what is accessed. In C there are pass by reference and pass by value, the difference between this is that pass by reference passes the variable memory location, while pass by value does not. This ties in which how programs are blocked and how they deal with scoping. Variables and values that are declared in a local scope cannot be access outside of that scope. Another controlling factor is how much space is allocated for programs. Programs such as Fortran77 are allocated all their need memory load time, and the location of variables are fixed to its location for the duration of the program. This is called a static environment. This type of environment does not take recursion as well as nested functions. In stack based environments, the record is created and added to the stack and when it is done it is removed from the stack. This means that programs are not allowed to statically allocated procedure blocks. However, this adds more to what needs to be kept track of, such as where the function was called, and a pointer to the current environment.

Overall there are many programming languages and they all differ but they all contain some sort of basic control. These controls can come in forms of loops, statements, and even lower level memory placement.