Bhavik Patel

CS:3820 Programming language concepts

Jeff Hajewski

10/13/2020

Midterm

The article “Fundamental Concepts in Programming Languages” by Christopher Strachey touches on many key concepts of programming language. The author talks about many of the programming language concepts, structure, and organization in greater depth and in an abstract manner. Some of the key concepts the author talks about in this article are assignment commands, L-values and R-values, definitions, types, expression, commands, values, and many more. The words that we use to describe the programming language are often misunderstood by different people. One of the misunderstandings is straight forward and muddling thinking and the second is being a difference in philosophical outlook between mathematicians. This difference may not be very large but they are good for attacking problems in the programming language. More than often problem-solving in the programming language requires calculus, but it is very much less to worry about because programmers attempt should be to formalize concepts and ideas to get the desired result. The words used in the programming language have different meanings associated with a programmer and mathematician, so it is important to clarify the concept.

The assignment commands, they are often used by the programmer to assign a specific value to the objects or variables. The assignment commands can assign any integer, text, or even mathematical formula to a variable or an object. Assignment command can also evaluate the expression and do the assignment. In contrast, you would not need any assignment commands if you were to write integer or text on paper. It can be just written without any assignment, but in the programming language, we have to refer to assignment commands to store the value in some safe place in memory so we can use it. Most programming languages use “=” as the assignment command.

L-values and R-values, these are part of variable and object assignment. The left side of the value refers to the location of the memory, and we can give it a name so we don’t get confused. It can be a character, word, complex, or precision number. Some of the locations are addressable and some of them are not because it depends on what the programmer is trying to do with the R-value assignment. The advantage of having an L-value is that we can check the stored value and update it. For example, int i = 2; this will store the R-value “2” into the location of L-value “i” somewhere in computer memory, and once it is assigned we can get that value or update it. Another concept “names” are important in the programming language because they work as an identifier for the R-values. We must give a variable or object name that signifies the R-value because giving an improper name to a variable or object in the programming world often leads to misunderstanding and can cause the program to crash. Another important type or concept of the programming language is numerals. The numeral can be a type of integer, double, decimal, or binary. Numeral can also be treated as a text string in many programming languages. The interpretation of the numerals which are also names are the single digits and they are constant with the appropriate R-value.

The expression is strings of commands that are executed one by one such as addition, subtraction, multiplication, or division. Expression commands are executed similar way to assembly language or machine language. To the large extent, the complexity of the commands has increased due to the higher level of programming which allows the right side of the assignment to be more complex and lengthy.

For example,

```
        LDI R16, 2    R16: 2

        LDI R17, 4    R17: 4

        ADD R16, R17    R16+R17: 6
```

The example above of the AVR code adds two registers into one that outputs the register R16 = 16. The code commands are executed line by line when ran, and that’s how the most expression is executed in the programming language. At the end of all the string expression execution will have the desired result of adding two numbers.

Value, it is a characteristic feature of the expression. Every expression holds a value of some sort that will be assigned to the L-value after the execution of the command. Just like the math, the right side is considered as the value of an expression, variable, or object. When computing the value from an expression we need to also have to take care of its environment components because some expressions are depended on the value of their environment components. For example, a + 2 + b + 5, where “a” and “b” are the environment of the expression. The expression is dependent on the value of a and b and if they are not defined in the program then command execution is likely to complain about it and the program will fail. Some languages allow us to define such an environment component by using “where” others you have to declare public or private variables before you write an expression.

Evaluation is another part of the programming language. The programming language evaluates the expression by identifying its different components. The evaluation process differentiates expression operators and operands to evaluate the expression. The evaluation process is designed as it applies from left to right meaning in math we follow the PEMDAS to evaluate basic math expression, so in the programming language, the same thing is done. The multiplication and division are performed before addition and subtraction. The evaluation also deals with conditional expression. The most often used conditional expression in the programming language is if-then-else. When executing this conditional expression evaluation process checks the condition one after another and if one condition is matched then the other condition becomes undefined and the result is returned.

Variable are part of every programming language. They store a value that can be changed or update during the execution of the program. There are two types of variables in the programming variables free variables and own variables. Free variables act like public variables that we often encounter in many languages like C++ or Java. They are declared as a public variable in the program so they are accessible anywhere in the program. Their value can be updated in program functions or classes. On the other hand, own variables act like private variables that we encounter in a most programming language. Their value can only be changed by a specific class or function. They often need a getter and setter to access and change the value. Fixed and free is a variable that remains the same thorough out the program its value is never changed or altered in the program such as pi, cos, or sin. They are constant variables that are available in the programming language. The constant variable value remains the same throughout the program execution. Their inaccessibility makes them safe from alteration except for the body of function or class.

Functions and routines are another way the expressions are written and executed in the programming language. We apply specific arguments in the function and produce the value. Most of us have written functions in our programming careers. The function can compute whatever we defined as it to do. Functions are there to remove redundant codes from the program so the code looks neat and organized. Function often used public and private variables to do certain tasks or computations.

Representation of functions is one of the key concepts of the programming language. When we create a function we need to be clear about L-value and R-value. We often return the left side with a computed value on the right side. We need to clear about what parameters we are passing in and what are we going to compute for the R-value. We need to be aware of our expression and required environment components otherwise the value computed would not be correct. In Java, we defined function as public calculate_sum(int a, int b); it is one of the ways we define and represent the function in Java. Every programming language has a syntax for defining a function. Representation of function is a key concept because functions can be private, public, or constant, and having a correct representation is important in the program to avoid confusion and failure. The function can also be defined recursively or non-recursively, and it is up to the programmer to decide which way is convenient for them and the program.

Today most of the programming language deals with many objects. Some of them are very similar to one another and some are not. Types can be represented as Integer, Float, String, Character, List, etc. Types of an object in the programming language determine its representation and constraints the range of objects is used to represent. Some of the types can be polymorphic meaning it can represent any type of object for one variable. For example, a = 1 or a = “one”. In the example, the variable “a” can either be represented as an integer type or string type, also the variable “a” will be called polymorphic because it can be assigned to any type of object. In some programming languages, the type is represented by the L-value for example in java int a = 0; initialize variable as a type of integer, and if you try to assign a string or some other type object to a variable then the compiler will complain about it.

List, most of the programming language has a type list or something similar to list like array or vector. The list data structure is used to store a large number of data in the program. The list either can be singly linked, doubly linked, or matrix. It depends on language design and architecture. The list can be processed with functions like insert, delete, get index, reverse, etc. Often computer programmer implements the list architecture with pointers or node in their starting of the college career. Nodes and pointers are formally used to implement the architecture of the list or binary trees. The programmer either defined the node type or create their own to implement the list or binary tree. The components are basic for example the Java linked list node object contains head and tail parameters where the head contains the value and tail points to the next node head and these allow us to create the linked list architecture. The implementation of the linked list can be different but it’s one of the ways to get the point across to the audience.

There are various other types of data structures in the programming language such as tuple, array, vector, sets, maps, etc. Some of them behave similarly to the list structure and some of them work as sets of two values. They are also key aspects of the programming language because they are already built into the language as a feature so programmers don’t have to implement them which makes it convenient.

In the end, if we combine all of the concepts discussed and described then it would result in the programming language. The programming language as mentioned before is the execution of strings of expression. All of the data types, variables, values, and data structures work together to create useful programming languages like Java, Python, and Haskell. Some of these concepts are hard to understand in abstract manners but once these concepts are learned and introduced through vivid programming language then they make a lot of sense. In conclusion, I would say that understanding concepts of programming language can lead to clarification and clarity on how a certain language is written and how it works the way it does. It allows programmers and mathematicians to resolve issues in a timely and convenient manner and allows them to get the best out of both.