Joshua Catoe

CSCI 350

Survey of Programming Languages

1) **FORTRAN**

FORTRAN (FORmula TRANslation) is a multi-paradigm programming language mainly aimed at scientists and engineers for mathematical and scientific computing. Designed by John Backus and developed by John Backus and IBM in 1957, its purpose was to make translating mathematical formulas and expressions into code easier and more efficient. In doing so, it became the first high-level programming language, making the jobs of programmers much easier. Some of its touted features are its advanced array handling, libraries for parallel computing such as Open MPI, and, being so old, its vast library of computational functions. The main criticism of FORTRAN is its age. It is generally looked upon as being old and antiquated, when in reality it is still being used in the mathematical and scientific communities, and is still being updated, with its latest release in November 2018.

**BASIC**

Designed and developed by John Kemeny and Thomas Kurtz in 1964, BASIC (Beginner’s All-purpose Symbolic Instruction Code) was a non-structured (i.e., no paradigm) language designed to give programming experience to a wide audience. Its main purpose was to teach computer programming to students outside of the science and engineering fields and although it was targeted at students, it spread to home computers a short time later. BASIC’s most praised feature was its simplicity. It had a more limited number of functions and simplistic, intuitive syntax that made it far easier to use than competing languages such as FORTRAN. Simplicity was also its biggest criticism, as some people felt that its non-structured and simplistic nature would lead beginning programmers to bad programming habits in the future.

**C**

C is a procedural paradigm language suited for general programming tasks as well as low-level activities. It was designed by Dennis Ritchie and developed by Dennis Ritchie and Bell Labs in 1972 for the purposes of allowing low-level access while still being a high-level language and high portability for cross-platform programming. The most important feature of C and what keeps it relevant, is its low-level access to memory, which enables manual memory management and the ability to use pointers. The biggest criticism of C is the freedom it offers the programmer. While having access to memory can make C more powerful than other high-level languages, it can also lead to more severe errors, should they go unnoticed.

**Prolog**

Prolog (Programmation en Logique) was designed and developed by Alain Colmerauer and Robert Kowalski in 1972 in Marseilles, France; it is part of the logical programming paradigm and is widely used in applications requiring logical decisions, most notably artificial intelligence. It was originally designed for natural language processing, that is, making computers understand natural languages such as French or English. Two of the main features of Prolog are backtracking and unification. Backtracking occurs when a query/task fails, and Prolog backs up to the previous query/task in order to find a new logical flow. Unification is the act of “unifying” one term with another; for example, if name(Judy) was a rule and the query name(x) was run, “Judy” would be unified with “x” and they would be treated as the same terms. Some of the criticisms of Prolog include not standardizing some features across Prolog implementations, which can lead to confusion in moving from one to another, and the fact that arrays do not exist.

**Python**

Python is a hugely popular, multi-paradigm, general purpose language that focuses on code readability and clarity. It was designed and developed by Guido van Rossum in 1990 as a successor to a language called ABC. Its features include those that can be found in other high-level languages such as easy-to-read syntax, large libraries and high portability, but what really sets it apart is its support for multiple programming paradigms. By default, it has support for procedural, object-oriented and functional paradigms, but supports many others through extensions. One of the biggest criticisms of Python is that it is slower than other similar languages, as it is interpreted not compiled. Other criticisms include Python’s dynamic typing instead of static typing, which many believe will lead to more bugs, and the use of indention for code flow instead of brackets, which is more a matter of taste than a real problem.

**Haskell**

Haskell is a functional paradigm, general purpose language that has mostly seen adoption in industry. It was designed and developed by a large group of people (too large to list here) in 1990, with the purpose being to create a more open-source functional language based on popular closed-source languages at the time such as Miranda. Haskell’s most unique feature is that it is a “purely” functional language, meaning that it has no side-effects. No side-effects means that once a variable has been assigned a value, it cannot be changed. Some criticisms of Haskell are its lazy evaluation which can lead to decreased performance in some cases and the fact that it has no side-effects can make certain tasks very difficult to complete, such as I/O.

**Java**

Designed and developed by John Gosling in 1995, Java is a widely used, multi-paradigm (though mostly seen as object-oriented) programming language that is aimed at high portability and security with its Java Virtual Machine (JVM). It was originally designed for embedded systems such as televisions and set-top boxes but grew to be a much larger general-purpose language. Java’s most unique feature is the JVM, a software-based platform that it runs on instead of the computer’s hardware that provides a high degree of portability; any machine that has Java installed can run a Java program, regardless of platform. The JVM also provides a layer of security, as it runs in its own sandboxed environment that can prevent malicious code from reaching the actual hardware. Some of the criticisms of Java include its comparatively slower performance, as the JVM and garbage collection add time to the program execution, and the security of the sandbox environment, as several exploits have been found through the years.

**Swift**

Swift is an up-and-coming, multi-paradigm language that is mainly used as the platform for creating applications for Apple operating systems. Introduced by Apple in 2014, Swift was designed to be a clean, modern alternative to Objective-C, which was the language that was previously used for development on Apple’s operating systems. Originally, it was closed-source but eventually became open-source and became available on other platforms such as Linux. The most important feature of Swift is that it works with Objective-C so that older applications still work and can be maintained. An unavoidable criticism of Swift is that it is a new language, which means limited libraries and tool sets for developers, but this will be less of a problem over time. Also, Swift only works with iOS 7 or later, however studies show that most iOS devices are running a version later than 7 anyway.

2) There are several similarities that exist between these languages. First, although the original BASIC is pretty much dead, and Prolog is not that popular, the rest of the languages are very much alive and well, even FORTRAN, which is still being updated and built upon. Second, all of these can be grouped into programming paradigms; FORTRAN and C would be grouped together as procedural, Haskell as functional, Prolog as logical, Java as object-oriented, etc. Lastly, most of the newer languages were inspired or based on the older ones. For instance, Java was inspired by C and even uses a lot of the same syntax, Swift was inspired by many languages on in the list and incorporates ideas from them, and FORTRAN was probably an inspiration for every language that came after it.

3) For my next job, I would most like to use Haskell. Functional languages have interested me for a while now, mainly in how different the thought process is from the procedural and object-oriented styles that I am used to. Also, I feel that knowing a more specialized language like Haskell would set me apart on the job market, and hopefully lead to higher pay.