Eduardo Pantoja, Trevor Osborne, & Bo Mortenson

CSCI 305 - Concepts of Programming Languages

Language Workshop Paper - Go

What is Go?

Go, also known as Golang, is a statically typed programming language developed in collaboration by Rob Pike, Robert Griesemer, and Ken Thompson at Google. When we were originally thinking of which language our group should choose for our Concepts of Programming Languages project, Trevor Osborne suggested that we do our paper, presentation, and workshop on a more recently created language. After some debate, we decided on the fairly recent addition of Go. With how many languages boomed before the year 2000, with several of these older languages still being used today, we figured that many of the groups would choose older languages as there simply is a larger support and usage for them. After looking into the language more on our own time, it was a unanimous decision amongst the group to go for Go.

Historical Background of Go

In 2009, Rob Pike and his colleagues got increasingly frustrated with how slow and inconvenient software development had become at Google. Many people at Google struggled to get work done productively with C++, with some problems taking an entire day to solve. Taking inspiration and features from languages like C, Pascal, and Oberon, Go was created to make developing easier for readers, writers, and debuggers on both large and small scale projects (Khrupa, 2018).

A lot of debate on the name of this programming language has come up since its original inception. Both the name “Go” and “Golang” are used very commonly when referring to this language. Google has officially stated that the name of the language is indeed “Go”, however both names are used interchangeable among many (Boyd, 2021). This confusion has sprung from the name of the website that hosts the Go programming language, known as golang.org. However, this argument is not really valid since the name for this website is simply golang.org due to go.org being already taken when being created. For simplicity, we will be referring to the language as “Go” for the rest of the paper and during the presentation.

Since Go’s initial launch, the language has gone through many updates and changes. One of the more substantial updates was Golang 1.5. Releasing in August 2015, the compiler toolchain completely removed any trace of C and converted to Go completely. This resulted in Go becoming an even easier language to program in and made it more self-sufficient. Since this was the ideal state in the eyes of the creators, future updates strived to improve smaller features like tools, libraries, frameworks, and etc. (Khrupa, 2018).

Problems Go Solved & Key Features

Go was built as a replacement for high-performance server-side languages like Java and C++. Go is commonly used today in cloud and server-side applications, command-line tools, artificial intelligence, and data science. It was a big success amongst Google developers for its extreme simplicity (Khrupa, 2018). As the language expanded to other companies, many developers agreed that the language was easy and simple to pick up for not just expereinced developers, but new ones as well (Boyd, 2021).

Some of the biggest features that Go provides is that it is an open-source language, that it is statically typed, features a fast automation process, and is a compiled programming language. Go features an incredibly powerful garbage collection system, similar to Java, that allows developers to not worry about freeing up pointers and creating dangling pointers. Some of the biggest companies that use Go include YouTube, Apple, BBC, Facebook, and several more (Sharma, 2021).

Pros of Go

Go is a very easy language to pick up on its own right, but it is even easier if you are already familiar with the syntax of C and C++ (Sharma, 2021). The language is designed to work on projects that are Google-sized, so there are no issues when used on other large-scale projects (Boyd, 2021). This does not mean that the language is exclusively restricted to large-scale projects however, it is still very commonly used in a wide range of project sizes. Go also allows developers to work with other platforms such as Windows, Unix, Linux, and BSD. Go features no virtual machine, all the code is compiled directly to machine code which saves time on the compilation process. After the code is compiled, only one executable file is produced which does not contain any sort of dependency (Sharma, 2021).

Cons of Go

While Go features many appealing features like simplicity in syntax and improved productivity, it is still a very new language. This makes it hard for Go to grow and gain popularity *quickly* as many companies are still using languages that have more of a long-standing reputation such as Java and Python. Some of Go's most appealing features are also its own worst enemy; many developers report that the language is *too* simple. The simplicity of the language makes it nearly impossible to perform any sort of complex and abstract processes. Go also has a lack of support for GUI libraries. This really hinders the language from its competitors of Java, Python, and many other languages supporting these features. Go also has poor error handling as many developers are forced to manually catch all errors using try statements and other built in error catching tools provided with the language (Sharma, 2021).

Go vs. Python

Since Python is a language that we will be using to complete our Lexical Analyzer project, let's compare how these two languages compare to one another. Python already has a great standing reputation amongst coders and non-coders, which Go strives to obtain that reputation as well. Python is primarily used in areas of data science while Go is used for system programming. Since Python is dynamically typed it can be easier to use than Go for quick prototyping. As mentioned above, Python also contains GUI support natively while Go does not (Sharma, 2021). As of right now, a lot of new programmers are starting with Python but as the years go on, Go is gaining more and more popularity.

Our Personal Experience with Go

In our own personal experience, Go was a very familiar language to learn. At Montana Tech, languages like C++ and Python are heavy focuses in our curriculum. With formatting similar to that of C++, with a looseness like Python in declaration, it felt very at home to use.

The only strange syntax that may be awkward for new users to get used to could be the setup for methods to make variables public access. Instead of declaring them "public" like you would in C++, you detonate a public variable by capitalizing the first letter. This does tell you instantly looking at the variable that it is public, but initially this may be a strange concept to grasp. This does change some formalisms that users may be used to as well. Generally constant variables are defined by making all their letters capital, this is not a requirement, but is generally expected formatting in most languages. This can only be done if you specifically want the constant variable to be public access, if you need the variable to be private, this formalism cannot be used.

Something that users will catch onto quick, but can seem strange as well, is the setup of variables. In most languages, the type of variable is defined first. However in Go, it is always defined second. This can be avoided all together with Go's ability to predict a variable's type similarly to python. This is handled very specifically, however, using a colon before the equals sign to denote the flexibility in that variable's type. Overall, Go provided an easy and familiar experience all around. Pulling great elements from several languages lends itself into being a great language for experts and beginners alike.

Conclusion

Go is a rising and upcoming language that is increasing in popularity each year. With the goals set by Rob Pike, Robert Griesemer, and Ken Thompson, the language has greatly succeeded in what they attempted to resolve. Even though Go was initially created to improve productivity at Google, it has expanded to many other companies that greatly benefit from its simplicity. Taking inspiration from popular and vetted languages like C, C++, Pascal, Oberon, it provides a familiar home to veteran programmers and is even friendly to new programmers. While Go has some cons that hinder the language, it has many pros that prove the language can strive even in its infancy. Go still has a long road ahead of it, but it will no doubt continue to make positive strides and improvements as the years go on.

References

Boyd, W. (2021, May 25). What is GO? An intro to Google's Go programming language (aka golang). A Cloud Guru. Retrieved from https://acloudguru.com/blog/engineering/what-is-go-an-intro-to-googles-go-programming-language-aka-golang.

Khrupa, A. (2018, March 20). Golang: How it started - a history of success - qarea blog. Qarea. Retrieved from https://qarea.com/blog/the-evolution-of-go-a-history-of-success.

Sharma, A. (2021, September 6). Guide on go programming language. Appinventiv. Retrieved from https://appinventiv.com/blog/mini-guide-to-go-programming-language/.