All average: neither good nor bad. – The case of a contemporary corporate language called Go.

Camille Akmut

Abstract
Political and technical notes about a contemporary programming language of distinctive corporate allures, identity.
Go programming language

"Go is basically C with garbage collection (and concurrency) that’s not Java" - I wrote in my notes.

It further shares with Java its distinctive corporate allure, identity. Sun/Oracle then, Google now.

For obvious reasons this is not the kind of description that will be found in materials for this language, whose status is often uncertain (between marketing and pedagogy, somewhere):

Euphemisms like “Go is sometimes described as a "C-like language"” can be found in the reference manual, or textbook for this language instead, for example.

(The Go Programming Language by Donovan/Kernighan. Going forward we primarily refer to it, unless otherwise noted.)

From C, Go inherits pointers (operators * and &); but, one doesn’t have to do everything manually e.g. freeing memory, malloc and all the related ‘business’ from C [2] [3]

It doesn’t have classes (like Pascal, of which it claims to be a descendant), and uses struct’s like C instead (handled as part of chapter 4, “Composite Types”).

Then there’s interfaces (ch. 7), and I suspect they may be used for similar purposes: there’s talk of type interfaces and instances thereof (Haskell does this better, they were already in Pascal?).

A graphic (tree-like) representation of the genealogy of the language can be found in a figure of the preface on p. xii, “The Origins of Go” — drawn from an unnamed 2009 publication by the designers of the language. Accompanying descriptions offer further insights.

It puts an emphasis on concurrency (as most modern languages should, probably), with its “goroutines” (ch. 8).

“If you have used operating system threads or threads in other languages, then you can assume for now that a goroutine is similar to a thread” - the authors note much later (p. 218).
A discussion of threads and C can be found in 21st Century C, for example: “A single-threaded program doesn’t make full use of the resources the hardware (...). Fortunately, it doesn’t take much to turn a program into one with concurrent parallel threads - in fact, it often only takes one extra line of code.” (ch. 12, “Parallel Threads”)

The simplest representation of which is:

```plaintext
f()    // call f(); wait for it to return
go f() // create a new goroutine that calls f(); don't wait
```

(p. 218)

Its theoretical backgrounds are “communicating sequential processes (CSP) from Tony Hoare’s seminal 1978 paper on the foundation’s of concurrency.” “But Hoare’s CSP was a formal language for describing the fundamental concepts of concurrency, not a programming language for writing executable programs.” (p. xiii).

They most likely mean “Communicating Sequential Processes” [4].

**Compared**

Haskell is what happens when some of the smartest people are put together in the same room (albeit even figuratively).

Go is what happens when a company throws a lot of money at (previously respected, or respectable) computer scientists. The former got their namesake language, the latter got their checks, salaries.

It is neither a good language nor a bad one.

The scale being:

— Java being a fairly horrific language in my opinion, but a horror you can get used to like almost anything else in life, I guess (I wouldn’t know since I refused to get used to it). Responsible for turning away from programming and computer science untold amounts of people (the poor ‘Java generation’, as it will be remembered), to the detriment of that science.

P.P, pronounced in fact /piː/, a language so bad it has been variously described “a fractal of bad design” [3] — meaning errors are reproduced infinitely in lower levels — or a “random number generated language“ (but, I can’t find the source for this anymore). It appears in no academic discussion of programming languages I know. (No one should learn this non-language.)
— On the other end of the scale there are languages like Haskell and (((some argue))) Lisp/Scheme.

— In the middle lies Python, a language that makes everyone happy (but cannot be used everywhere unfortunately, for now).

Meanwhile Rust is a better Go.

Who wants to have “blank identifier(s)” (p. 12), i.e. _ _, all over their code? This is confusing, and seems to be an oddity of the language.

They made design decisions and suddenly found themselves stuck with it, and them? Who knows...

The ability to not be able to compile code with unused functions makes it a nightmare to debug (and not a virtue, as they propose). The proposed remedies are awful.

Go:

```go
package main

import "fmt"

func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}

func main() {
    fmt.Println(fact(7))
}
```

https://gobyexample.com/recursion

Rust:

```rust
fn factorial(i: u64) -> u64 {
    match i {
    0 => 1,
    n => n * factorial(n-1)
    }

```
Haskell:

```haskell
factorial :: Int -> Int

factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Compared to David Turner’s creation (Miranda) and its successors (e.g. Haskell), Thompson et al.’s effort feels a bit like college basketball, like most other ones.

David Turner is perhaps the greatest language designer to have ever lived; and to have made an equally unfortunate decision with licensing...

Greed is every Man’s downfall, as he learned again after all of his many predecessors had, and as will again his successors, present and future.

**References**


[2] “The `&` operator yields the address of a variable, and the `*` operator retrieves the variable that the pointer refers to, but there is no pointer arithmetic.” (p. 24) See also 2.3.2, “Pointers”.
