Building a programming language without the dependency on English

CMP6102 Individual Project: Project Mini-Proposal

Pandelis Zembashis — S15101590
Supervisor: Emmmett Cooper

October 2018

1 Inspiration

Computer languages are heavily westernised as the majority of early innovation around computer science, in the late 50's to 60's took place in the west (North America and Europe). Most if not all modern programming languages can trace their routes back to two of most influential languages of this time period, ALGOL 68 and COBOL (Nofre, 2010). Both of these languages were invented in a time where all interested parties were English speaking and were inventing processes that would go on to be used by parties that could communicate in English at some level. Not to mention that the most common input to this day is still the qwerty keyboard which is comprised of latin characters.

These early languages have grandfathered in English and the Latin character set as the basis for all programming languages we see today. This is incredibly limiting to the creativity of a given language, and the accessibility to non native English speakers or different cultures. We have limited ourselves to the symbols available to a character set that was not designed with the intention to be used to program.

2 The Idea

I propose a language be designed around its own character set without the dependency on English at its core. There are alternate writing systems that use logographic character sets where in which a single character can describe an entire word or phrase, take Chinese and its derivative script Japanese Kanji or Egyptian hieroglyphs for example. Using this logographic system as inspiration for a language opens up many more creative avenues for how the language can be designed.

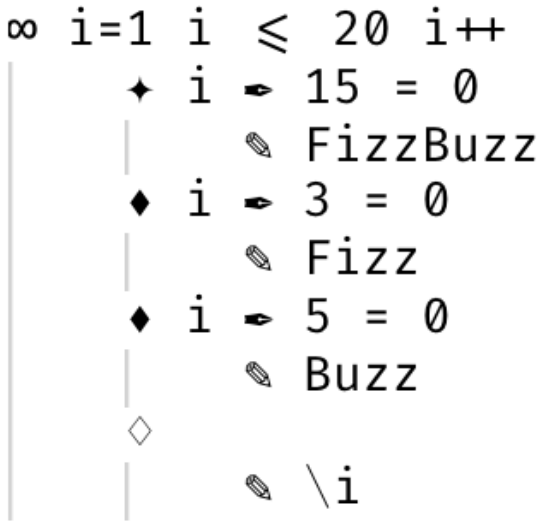
We can assign any meaning we wish to an arbitrary character instead of taking a complex set of English words or symbols to create language keywords we can have language symbols dedicated to each task. Similar to how some programming languages allow for macros, a single instruction that expands automatically into a set of instructions to perform a particular task, we could assign complex behaviour to single symbols.

Take this example of a fizz buzz program in a traditional language like Javascript 1 and pseudocode of a made up logographic language 2. In this program any numbers divisible by 3 return Fizz, 5 return Buzz and if divisible by both 3 and 5 it returns FizzBuzz.

Figure 1: Example of fizz buzz program in Javascript

```
for (var i=1; i <= 20; i++)
{
    if (i % 15 == 0)
        console.log("FizzBuzz");
    else if (i % 3 == 0)
        console.log("Fizz");
    else if (i % 5 == 0)
        console.log("Buzz");
    else
        console.log(i);
}
```

Figure 2: Pseudocode example of a logographic approach



3 Project rationale

I believe that the long lineage of languages being developed on top of a latin character set has lead to a stagnation in the innovation of languages. Most languages look very similar which is great for programmers who understand one modern language. It makes picking up a new language pretty easy when you understand one.

However this dependence on English keywords makes learning your first programming language disproportionately harder for those without a firm grasp of the English language. This makes learning to program much harder and sometimes inaccessible to other cultures and languages (Kamal et al., 2014).

By creating this language I hope to make a language that is accessible and sensible to users without previous programming experience. By utilising symbols users only need to learn the meaning of a symbol and not concern themselves with the difficulty of understanding the characters or cultural nuances (Eastman, 1982) of another language.

4 Goals and vision

As a bilingual Computer Science student myself I have experienced the difficulties understanding and adapting to various programming languages first hand. I also have an avid interest in teaching people how to code. As a founding member of the BCU Hackathon and Computing Society (*HaCS - Hackathon and Computing Society at BCU* n.d.) I have ran many programming workshops for first year computer science students over the last two years. This year I am also a teaching a group of firs time coders how to code through Code First Girls(*Code First* n.d.), they aim to teach girls with no previous programming experience how to program. I therefore have lots of first hand experience with the difficulties of both teaching, understanding and learning programming.

Using this experience I hope to put together a language that is more accessible to newcomers and can be used as an educational tool. While developing the language I can also test attempt to teach it to users and see how they react to it and adapt the language during the development process.

4.1 Project plan

To achieve these goals I hope to create an initial prototype that takes a subset of an existing language like Javascript and transpile from the logographic language to a that language to be executed. This will allow me to focus on the designing of the language and quickly prototype a set of symbols that work well together. During this stage I can use existing language tooling such as Babeljs (*Babel · The compiler for next generation JavaScript* n.d.) to try out different language semantics. This will enable me to put a working prototype of the language in front of users in order to validate weather or not the language functions well for its intended purpose.

With a firm set of commands and a basis for the design of the language I can then look towards creating tooling around the language to make it easy to develop in. Finally depending on the complexity of the first two stages I hope to be able to then create a compiler or virtual machine for the language to run in. I can use an existing compiled language such as GO or a virtual machine like the JVM to achieve this last stage of developing the language.

References

Babel · The compiler for next generation JavaScript (n.d.). URL: https://babeljs.io.

Code First (n.d.). *Code First: Girls*. en. URL: http://www.codefirstgirls.org.uk/.

Eastman, Caroline M. (Dec. 1982). "A Comment on English Neologisms and Programming Language Keywords". In: *Commun. ACM* 25.12, pp. 938–940. ISSN: 0001-0782. DOI: 10.1145/358728.358756. URL: http://doi.acm.org.ezproxy.bcu.ac.uk/10.1145/358728.358756.

HaCS - Hackathon and Computing Society at BCU (n.d.). URL: https://hacs.tech/.

Kamal, A. et al. (Dec. 2014). "ChaScript: Breaking language barrier using a bengali programming system". In: *8th International Conference on Electrical and Computer Engineering*, pp. 441–444. DOI: 10.1109/ICECE.2014.7026875.

Nofre, D. (Apr. 2010). "Unraveling Algol: US, Europe, and the Creation of a Programming Language". In: *IEEE Annals of the History of Computing* 32.2, pp. 58–68. ISSN: 1058-6180. DOI: 10.1109/MAHC.2010.4.