Building a programming language without the dependency on English

CMP6102 Individual Project: Project Proposal

Pandelis Zembashis — S15101590
Supervisor: Emmmett Cooper

November 2018

1 Introduction and background

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 Aim

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.

3 Objectives

The main purpose of the language is to make it more approachable for new learners with no previous programming experience and crucially, little to no knowledge of English. To achieve this the basis and most crucial objective for the language will be to use no English based keywords whatsoever in the design of the language. Any such keywords go against the whole purpose of the language. This will be measured by checking the language specification once it has been finalised and checking that there are no keywords based on or derived from English.

The next most important function of the language should be to be powerful enough to learn all major elements of programming. A measure for this would be for the language to be Turing Complete. This is to say that the language is as computationally powerful as a Turing machine (Turing, 1936). This would enable users of the language to solve any computational problem within the language.

The language should be easy and quick to get programming in. It would be foolish to create a language designed for learning that is incredibly difficult to install or configure. To achieve this there should be well designed tooling built around the language to assist users with getting set-up. This could include things like editor integration through a plugin, and easy installer or a completely web based UI so that there is no installation needed. This can be measured by exposing a group of users to the language and gathering feedback. The group can be comprised of first time coders, beginners and intermediate skilled coders. The feedback can be gathered as a survey from the users and by watching over the users and making notes of their performance. If users are able to create a basic program with little to no guidance then the objective of being easy to start programming in would be complete.

4 The Product

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.

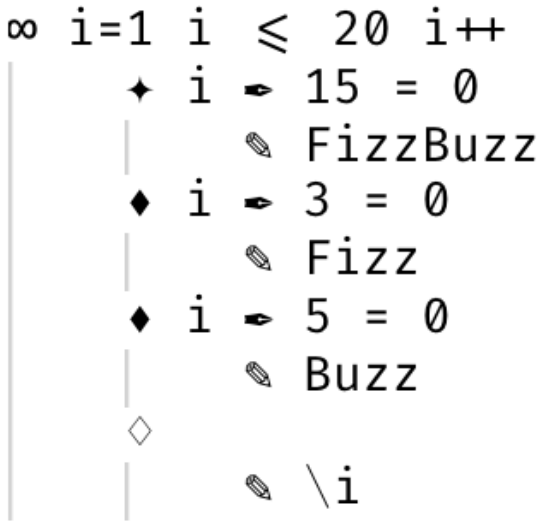
We can assign any meaning we wish to an arbitrary character instead of taking a complex set of English words or symbols to create language keywords we can have language symbols dedicated to each task. Similar to how some programming languages allow for macros, a single instruction that expands automatically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



5 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.

The end product does not aim to compete with any traditional language on features, speed or security like a language traditionally would. Therefore

the focus for this product is not the same as a traditional language and aims to experiment in different ways. By working outside the traditional requirements for a language we can tackle problems that have yet to be looked at.

6 Methodology

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.

7 Schedule

Due to the unique challenges associated with creating a new programming language the process needs be split up into manageable sections.

For the initial development phase I will be creating a series of minimum viable products (MVP's). This will allow me the time and flexibility to experiment with Implementation methods. At this stage of the project there isn't a clear best approach. As my first MVP I will attempt to create a transpiler that will transpile to JavaScript. Depending on the difficulty and success of this MVP I will then attempt to create an interpreted or VM based language.

Before getting to this stage however I will need to learn about transpilers, interpreters and any other alternative methods of creating a language. This

will be carried out during an initial research phase and then continued in parallel with the MVP development process. During the MVP process there will be allot of issues to overcome so it is important to schedule in research time within this phase.

Another step to come just before the initial MVP stage is to clearly define the symbols to be used in the language. This is a crucial part as it defines the whole language and all subsequent steps.

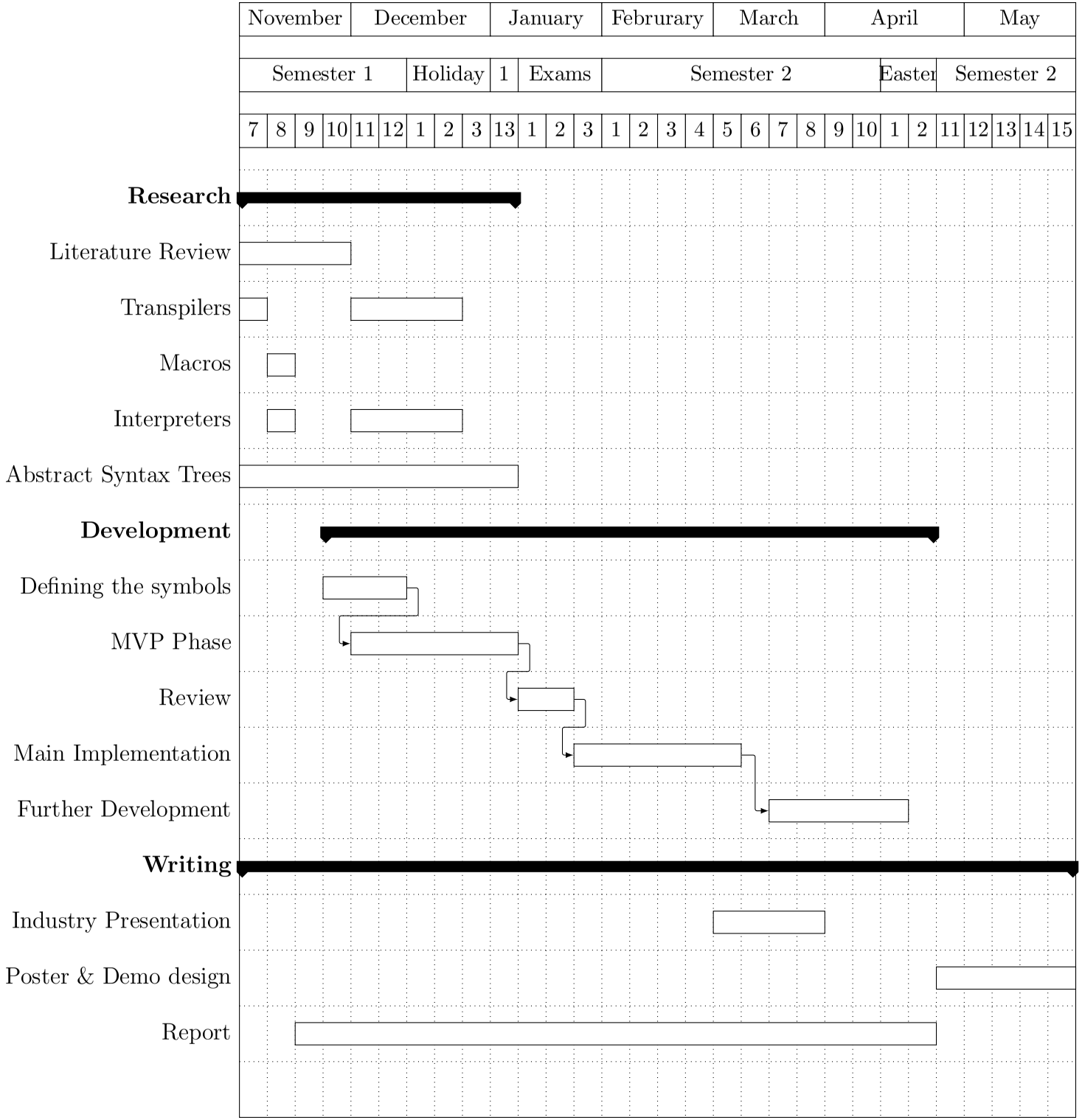
Following the development of the MVPs there will need to be some time to review the success of each method. This stage will determine how the final implementation will be carried out. The most successfully implementation, transpiled, interpreted or vm based will be taken further and made fully featured.

During the main implementation period is when the tooling and ease of use of the language will be addressed. Depending on time constraints during this step there will be a browser based interface created to aid in the ease of use and to achieve the goal of being able to run without installation.

The main implementation section extends up until the Industry presentation where I will be able to gather some feedback. Some of this feedback can then be worked into the further development stage. This stage then continues up until the report deadline.

In parallel to all of this the report is also being written and pieced together, detailing the successes and failures of the MVP stages, the research and the whole design and implementation of the language.

Bellow is a gantt chart which shows the schedule as individual blocks of 1 week. Each week is numbered with the matching semester week. Those that are not within a semester, *Christmas Holidays*, *Exams*, and *Easter* have their own week numbers. The chart starts from Semester 1, commencing from Week 7.


November
December
January
Februrary
March
April
May
Semester 1
Holiday
1
Exams
Semester 2
Easter
Semester 2
7 8 9 10 11 12 1 2 3 13 1 2 3 1 2 3 4 5 6 7 8 9 10 1 2 11 12 13 14 15
Research
Literature Review
Transpilers
Macros
Interpreters
Abstract Syntax Trees
Development
Defining the symbols
MVP Phase
Review
Main Implementation
Further Development
Writing
Industry Presentation
Poster & Demo design
Report

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.

Turing, Alan M. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* 2.42, pp. 230–265. URL: http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf.