Language Syntax

Non Technical terms

As the domain of the language is education, the syntax should be as accessible as possible. The fact that we are not tied to the English language allows us to release ourselves from historical terms for different elements of the language, there is no longer a *traditional* term for anything we are doing, other than the english literal translations of the usual terms.

This means that we can use more accessible, non-technical terms to describe elements of our language. For example the word *gníomh* can be used in place of the term function, *gníomh* directly translates to an "action". Instead of referring to a single file as a "program" or "script" we can refer to it as a "story" (*scéal*) etc. Programming terms have a tendency to be very technical, leaving them cryptic for beginners, for example String, Floating point. These mean nothing to a beginner, we have no need for these terms in this language.

Syntax family

This language will base many of it's syntactic features from the family of languages with C-style syntax. However, many of it's syntactic features will be derived from the linguistic structure of the Irish language itself.

Example

Below is an example script in the language. We create a class (*creatlach* meaning skeleton) for a person (*Duine*), then a class for a man (*Fear*) that inherits from *Duine*.

Fear has a constructor that takes a name (*ainm*) and defines an action (*gníomh*) called *abair* (speak) that prints a string containing the name.

Duine defines a function *siúl* (walk) that takes in an argument *céimeanna* (steps) and loops from 0 to *céimeanna*, printing the string "*Ag siúl*" (walking) each time.

```
creatlach Duine {
    gníomh siúl(céimeanna) {
        má céimeanna == 0 {
            scríobh("Ní gá dom siúl")
        }
        le céim idir (0, céimeanna) {
            scríobh("Ag siúl")
        }
    }
}

creatlach Fear ó Duine {
    nua (ainm) {
        [ainm seo] := ainm
    }

    gníomh abair() {
        scríobh(ainm + " is ainm dom agus is fearr mé")
    }
}

sean := Fear("Sean")
[abair sean]()
```

Structures

Expressions

Expressions can be formed from the following operators, with usual operator precedence rules

Numeric operators $(\mathbb{R} \times \mathbb{R} \to \mathbb{R})$

+	−	*	/	%
plus	minus	times	division	modulo

Boolean operators $(\{\text{fír}, \text{bréag}\} \times \{\text{fír}, \text{bréag}\} \to \{\text{fír}, \text{bréag}\})$

&	|
and	or

Both operators are short-circuiting

Comparison

==	!=	<=, >=, <, >
equal	not equal	order comparison

Comparison is strict, for values to be compared they must be of the same type.

Variables and scoping Variables are lexically (block) scoped.

Variables are defined by the definition statement

```
<id> := <expr>
```

Variables can be reassigned by assignment statement

```
<id> = <expr>
```

Conditionals Setanta has one (so far) conditional control flow structure, the if (*má*) statement

```
má <expr> <stmt> nó <stmt>
```

The *nó* branch is optional

Looping structures Setanta has two loop structures (so far)

- Range loop: `le <id> idir (<expr>, <expr>) <stmt>`
  Loops between the first and second expression, tying the result to `<id>` on each iteration
- While loop: `nuair a <expr> <stmt>`