Birch Language Specification

June 16, 2024

1 Purpose

Birch sets out to be a language that merges the best of functional and OOP into one style, rather than merely supporting both paradigms, cutting down the breadth of possible ways to write a given piece of code. Statically typed, AOT compiled, and as minimal as possible while being reasonably safe.

2 Overview

Modules Modules define all structural scope in Birch. Any Birch file is a module, and any public module may be imported by any other module using the import keyword. Modules also can define data structures, which is dicussed in User Defined Types.

User Defined Types

Modules Any module may have an internal data structure defined, which will always be an algebraic type. If a module has a data structure, it may be instantiated. (If tuples removed, then modules will need destructuring syntax)

Tuples (and Structs?) Tuples may be prototyped, and elements given names. A function returning a tuple, may also give names to the elements of the tuple, without prototyping the tuple ahead of time. Elements may be accessed by name or index. Tuples may be automatically destructured at which point there will be no (Depending on how stream lined modules become, perhaps tuples will be removed entirely)

Functions

3 Tokens

3.1

3.2 Key Words

let	LET
priv	PRIV
vis	VIS
use	USE
pub	PUB
if	IF
else	ELSE
match or case or mux tbd	
for	FOR
while	WHILE
mod	MOD
data or type tbd	
to	TO
as	AS
unsafe	UNSAFE
None	
self	
Self	

3.3 Operators

$. . = + - */|\& (opfromabove) = <, >, = (opfromabove) =$

3.4 Symbols

[]()'" ; ?_

3.5 Types

u(8 | 16 | 32 | 64)	UINT_num
i(8 | 16 | 32 | 64)	INT_num
f(8 | 16 | 32 | 64)	FLOAT_num
usize	USIZE

3.6 Constants

[1-9][0-9]?	CONST_INT
i8-32	INT_num
f32-64	FLOAT_num
usize	USIZE

4 Grammar

prog mod_list main decl_list decl decl_list decl