Birch Language Specification

June 26, 2024

1 Purpose

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 Key Words

`use`	USE
`let`	LET
`priv`	PRIV
`vis`	VIS
`pub`	PUB
`if`	IF
`else`	ELSE
`match or case`	
`for`	FOR
`while`	WHILE
`mod`	MOD
`type`	TYPE
`to`	TO
`as`	AS
`unsafe`	UNSAFE
`self`	
`Self`	

3.2 Symbols

`+`	ADD	
`-`	SUB	
`*`	MUL	$+ - */|\& \; (op from above) = <, >, = \; (op from above) = [] ()'" \; ; ?\_$
`/`	DIV	
`&`	AND	
`|`	OR	

3.3 Types

`u(8 | 16 | 32 | 64)`	UINT_(num)
`u(8 | 16 | 32 | 64)`	INT_(num)
`u(8 | 16 | 32 | 64)`	FLOAT_(num)
`usize`	USIZE

3.4 Constants

`[0-9]*`	CONST_INT
`[0-9]?.[0-9]*`	CONST_FLOAT

4 Grammar

prog mod_list main decl_list decl decl_list decl