

Oat v.1 Language Specification

1 Grammar

The following grammar defines the Oat syntax. All binary operations are *left associative* with precedence levels indicated numerically. Higher precedence operators bind tighter than lower precedence ones.

$prog$	$::=$	$prog$
		$decl_1 .. decl_i$
$decl$	$::=$	global declarations
		$gdecl$
		$fdecl$
$gdecl$	$::=$	global variable declarations
		$global\ id = gexp;$
arg	$::=$	arg
		$t\ id$
$args$	$::=$	args
		$arg_1, .., arg_n$
$fdecl$	$::=$	function declaration
		$retty\ id(args)\ block$
$block$	$::=$	blocks
		$\{stmt_1 .. stmt_n\}$
t	$::=$	types
		int
		$bool$
		ref
ref	$::=$	reference types
		$string$
		$t[]$
F	$::=$	function types
		$(t_0, .., t_n) \rightarrow retty$

<i>retty</i>	::=	void <i>t</i>	return types
<i>bop</i>	::=	* + - << >> >>> < <= > >= == != & [&] []	(left associative) binary operations multiplication (precedence 100) addition (precedence 90) subtraction (precedence 90) shift left (precedence 80) shift right logical (precedence 80) shift right arithmetic (precedence 80) less-than (precedence 70) less-than or equal (precedence 70) greater-than (precedence 70) greater-than or equal (precedence 70) equal (precedence 60) not equal (precedence 60) logical and (precedence 50) logical or (precedence 40) bit-wise and (precedence 30) bit-wise or (precedence 20)
<i>uop</i>	::=	- ! ~	unary operations
<i>gexp</i>	::=	<i>integer</i> <i>string</i> <i>ref null</i> true false new t [] {<i>gexp</i>₁, ..., <i>gexp</i>_{<i>n</i>}}	global initializers 64-bit integer literals C-style strings
<i>lhs</i>	::=	<i>id</i> <i>exp</i> ₁ [<i>exp</i> ₂]	lhs expressions

<i>exp</i>	<pre> ::= <i>id</i> <i>integer</i> <i>string</i> <i>ref null</i> <i>true</i> <i>false</i> <i>exp</i>₁ [<i>exp</i>₂] <i>id</i>(<i>exp</i>₁, .., <i>exp</i>_{<i>n</i>}) <i>new t</i> [] {<i>exp</i>₁, .., <i>exp</i>_{<i>n</i>}} <i>new int</i> [<i>exp</i>₁] <i>new bool</i> [<i>exp</i>₁] <i>exp</i>₁ <i>bop</i> <i>exp</i>₂ <i>uop exp</i> (<i>exp</i>) </pre>	<p>expressions</p> <p>64-bit integer literals C-style strings</p> <p>Explicitly initialized array Default-initialize int array Default-initialize bool array</p>
<i>vdecl</i>	<pre> ::= <i>var id = exp</i> </pre>	<p>local declarations</p>
<i>vdecls</i>	<pre> ::= <i>vdecl</i>₁, .., <i>vdecl</i>_{<i>n</i>} </pre>	<p>decl list</p>
<i>stmt</i>	<pre> ::= <i>lhs = exp</i>; <i>vdecl</i>; <i>return exp</i>; <i>return</i> ; <i>id</i>(<i>exp</i>₁, .., <i>exp</i>_{<i>n</i>}); <i>if_stmt</i> <i>for</i>(<i>vdecls</i>; <i>exp</i>_{opt}; <i>stmt</i>_{opt}) <i>block</i> <i>while</i>(<i>exp</i>) <i>block</i> </pre>	<p>statements</p>
<i>if_stmt</i>	<pre> ::= <i>if</i>(<i>exp</i>) <i>block</i> <i>else_stmt</i> </pre>	<p>if statements</p>
<i>else_stmt</i>	<pre> ::= ϵ <i>else block</i> <i>else if_stmt</i> </pre>	<p>else</p>