

Miranda Operators and other Lexical Notation

<code> </code>	Start of comment line
<code>%</code>	Start of directive line
<code>+</code>	Add operator
<code>-</code>	Subtract/negate operator
<code>*</code>	Multiply operator
<code>/</code>	Division operator
<code>^</code>	Raise-to-the-power operator
<code>~</code>	Logical NOT operator
<code>&</code>	Conditional AND operator
<code>\ </code>	Conditional OR operator
<code>=</code>	Assignment operator Equal-to operator
<code>~=</code>	Not-equal-to operator
<code><</code>	Less-than operator
<code>></code>	Greater-than operator
<code><=</code>	Less-than-or-equal-to operator
<code>>=</code>	Greater-than-or-equal-to operator
<code>.</code>	Function composition operator
<code>,</code>	Guard/Case specifier in conditional expressions
<code> </code>	Separator in list comprehension
<code>;</code>	Where specifier in list comprehension
<code>:</code>	Append-head operator ("cons")
<code>++</code>	List concatenation operator
<code>!</code>	Indexing operator
<code><-</code>	List comprehension generator
<code>::</code>	Function type specification operator Type query operator in interpreter ("has type")
<code>::=</code>	Algebraic data type specification operator
<code>==</code>	Type synonym specification operator
<code>abstype</code>	Abstract data type declaration
<code>with</code>	Abstract data type signatures declaration
<code>()</code>	Empty value in IO () type
<code>[and]</code>	List constructors; ", " as separator
<code>(and)</code>	Tuple constructors; ", " as separator Infix-to-Prefix constructors
<code>\$</code>	Prefix-to-Infix constructor
<code>' and '</code>	Literal char constructors
<code>" and "</code>	String constructors