Concepts of Programming Languages, CSCI 305, Fall 2020
Regular Expressions for Rust Programs, Sept. 21

Small Rust Programs

Helpers:
alpha → a-z | A-Z

binNonZDigit → 1
binDigit → 0 | binNonZDigit
octalNonZDigit → binNonZDigit | 2 | 3 | 4 | 5 | 6 | 7
octalDigit → 0 | octalNonZDigit
nonZDigit → octalNonZDigit | 8 | 9
digit → 0 | nonZDigit
hexNonZDigit → nonZDigit | a | b | c | d | e | f
hexDigit → 0 | hexNonZDigit
alphaNumeric → alphChar | digit
sign → + | -
space → 0x20
tab → 0x09
carraigeReturn → 0x0d
lineFeed → 0x0a
whiteSpace → space | tab | carriageReturn | linefeed
singleChar → alphanumeric | _ | / | * | !
stringChar → singleChar | . | ` | sign | (|) | { | } | : | ;
commentChar → alphaNumeric | _, ., `, ‘’, +, -, (,), {, }, :, ;, space (0x20), tab (0x09)

Comments are not tokens, however, they need to be recognized and thrown away.

Test files may have been created on DOS or Linux:

- A newline in DOS files is carriageReturn linefeed (0x0d 0x0a)
- A newline in Linux files is simply linefeed (0x0a)

comment → nonDocLine | nonDocBlock | docLine | docBlock

nonDocLine → / ∘ / ∘ commentChar* ∘ (0x0d 0x0a | 0x0a)

nonDocBlock → / ∘ * ∘
(commentChar | 0x0d 0x0a | 0x0a |
nonDocLine |
/ ∘ * ∘ (commentChar | 0x0d 0x0a | 0x0a) * ∘ * ∘ /)*
∘ * ∘ /

docLine → / ∘ / ∘ (/ | **!)** ∘ commentChar* ∘ (0x0d 0x0a | 0x0a)

docBlock → / ∘ * ∘ (* | !) ∘ (commentChar | 0x0d 0x0a | 0x0a) ∘ * ∘ /

Token Name	Regular Expression
(	(→ (
)	) →)
{	{ → {
}	} → }
:	: → :
;	; → ;
binNumber	binNumber → (sign | ε) ∘ 0 ∘ b ∘ (binDigit | binNonZDigit (_ | binDigit)* binDigit)
boolean	The Booleans true and false will be recognized as identifiers, and returned as a boolean token after a keyword lookup.
character	character → ` ∘ ( singleChar | space | \ ∘ t | \ ∘ r | \ ∘ n) ∘ `
decimalNumber	decimalNumber → (sign | ε) (digit | nonZDigit (_ | digit)* digit)
floatNumber	floatNumber → (sign | ε) (digit | nonZDigit (_ | digit)* digit ∘ . ∘ (ε | (_ | digit)* digit
hexNumber	hexNumber → (sign | ε) ∘ 0 ∘ x ∘ (hexDigit | hexNonZDigit (_ | hexDigit)* hexDigit)
identifier	identifier → (alpha ∘ (alphaNumeric | _) *) | (_ ∘ (alphaNumeric | _) ∘ (alphaNumeric | _)*
octalNumber	octalNumber → (sign | ε) ∘ 0 ∘ o ∘ (octalDigit | octalNonZDigit (_ | octalDigit)* octalDigit)
string	string → “ ∘ (stringChar | space | (\ ∘ (n | r | t | \ | ` | “)))* ∘ “

Keyword tokens:
bool, char, else, false, fn, if, float, for, int, let, loop, main, return, string, struct, true, type, where, while, value