

The scanner uses the data structures written in the previous lab, namely the symbol table and the underlying hashtable

Besides that, it uses a vector to store the tokens gotten from scanning:

- The first element is the type of the token (identifier, separator, etc...)
- the second is the pair of coordinates in the symbol table

The scanner takes track of the position in text with an index variable and the line and token number, by checking how many tokens have been processed and how many newline characters

The scanning always processes one token at a time. At first it takes care of the white spaces, so checking at the start of the current substring is easier with regex

Then firstly the constants are checked, this is done, so ambiguities are prioritized, hence easier to manage, like  $1+2$  is more easily handled as 1, + and 2 in this way.

- the number can check if the last element is also a number, and since with regex the whole number is checked, we don't need to check ahead in another step

The next thing which is checked are the tokens, so we know whatever remains are the identifiers

- here the operators and separators are just checked as they are, since their characters cannot be present at other stage, so there is no problem with overlapping with something

else in some cases

- the reserved words are checked with regex, so we can distinguish from identifier and reserved word, like `print` is handled as an identifier, and not as reserved word (`print`)

and identifier (`a`)

Ultimately we check the identifiers, and since we don't have to worry about reserved words still remaining for this token, we can skip that check here