The scanner uses the data structures written in the previous lab, namely the symboltable and the underlyi
ng hashtable
Besides that, it uses a vector to store to tokens gotten from scanning:
- The first element is the type of the token (identifire, 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 ch
ecking 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 and finite automata

Than firstly the constants are checked, this is done, so ambiguities are prioritized, hence easier to manag
e,
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 so it doesnt identify the operator + as a string
- since we check the integers with a finite automata which only gives back the first subsequence which is
accepted
    we need to look ahead to check if the next char is not a letter or digit, since that should give an error

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 presen
t
at any other stage, so there is no problem with overlappong with something else in some cases
- the reserved words are checked with regex, so we can destinguish from identifier and reserved word,
like printa is handled as and identifier, and not as reserved word (print) and identifier (a)

Ultimately we check the identifiers with finite automata, and since we don’t have to worry about reserved
words still remaining
for this token, we can skip that check here
- since the identifier can contain letters and digits we dont have the same problem as with integers