included. The single zip file should be uploaded to Canvas for submission; the code will be run through automated testing scripts and will also be examined during the scheduled final demonstration for marking.

If you are unable to complete any part for the whole SPL language it is suggested that you should reduce the language to a subset so that you can complete the compiler stages.

In order for you to test your compiler, some example programs written in SPL are supplied together with some skeleton files that might be helpful in building your compiler. These skeleton files will be explained in the lectures at an appropriate time. The files can be downloaded from in a from Canvas as a zip archive, or found on the U: network drive.

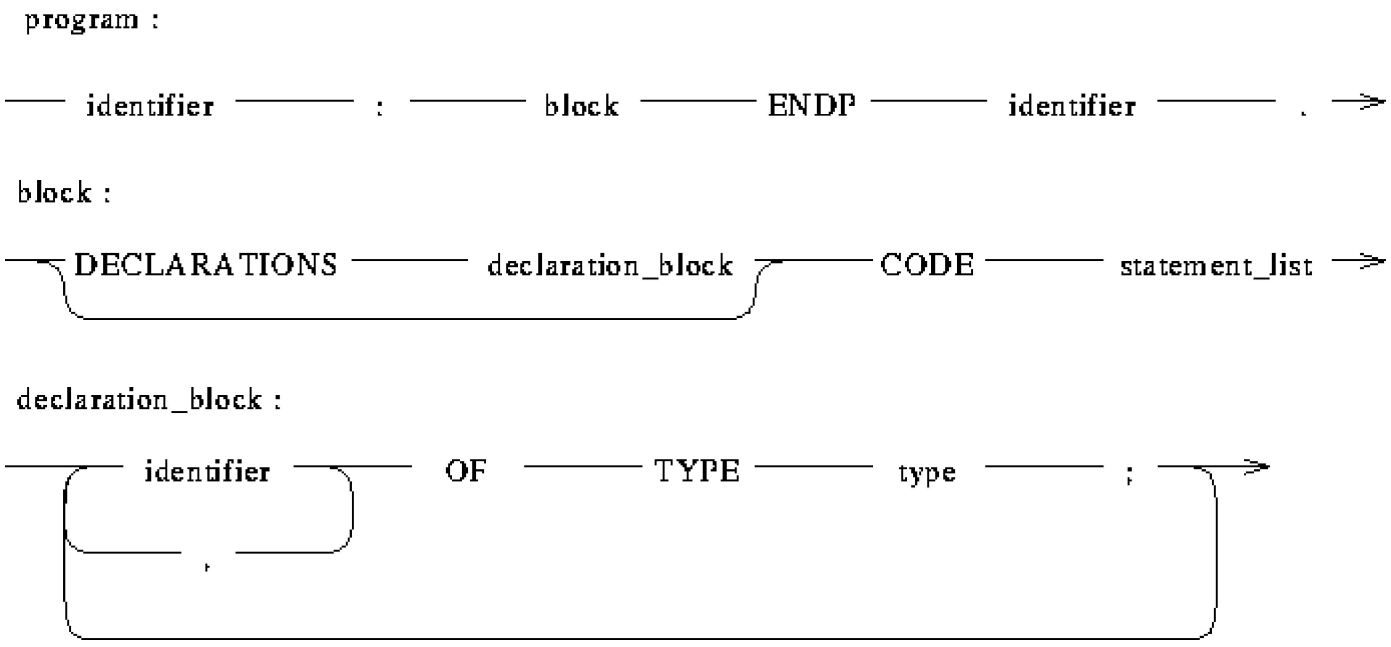
There are also video tutorials available to help you with this assignment on YouTube by following the link on Canvas.

SPL Language Description

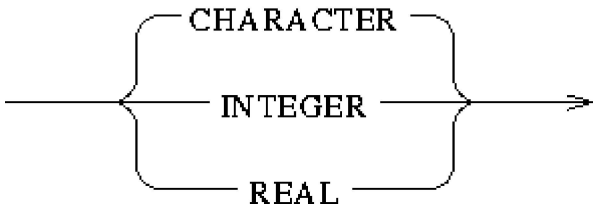
The SPL source resides in a single file. The following diagrams illustrate the complete structure of a SPL program. Identifiers follow the same rules as for PASCAL and other common languages. They must start with a letter and can only contain alphanumeric characters. Identifiers cannot be reserved words in the language.

Three data types are supported in the language, *characters*, *integers* and *reals*. Characters are always enclosed inside single quotes, 'a'. Integers and reals can both be either positive or negative. Reals contain a decimal point and must have at least one digit after the decimal point i.e. 3.0 is a valid real, 3. is not.

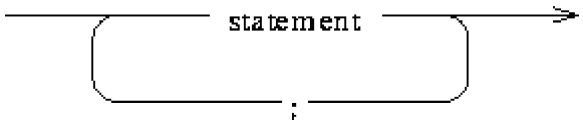
Programs are labelled with a program name, which must be a valid identifier. The identifier must be present both at the start and end of the program, and may not be used elsewhere in the program as a variable.



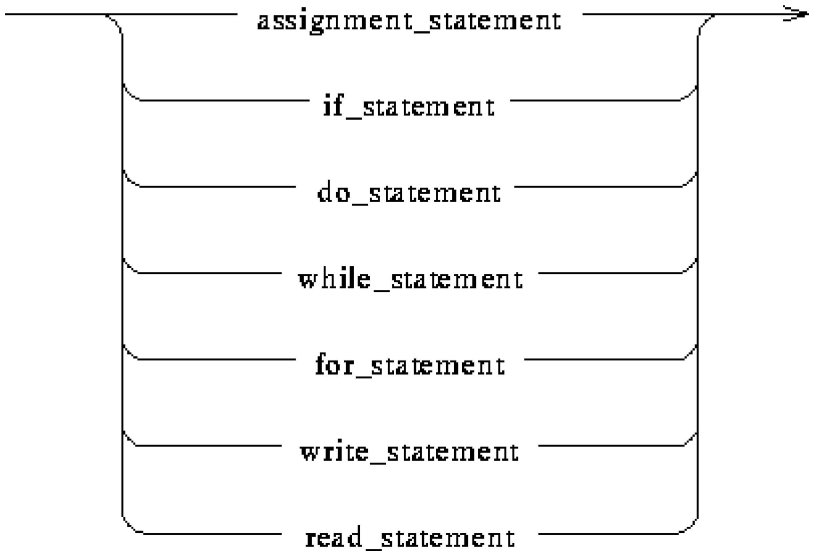
type :



statement_list :

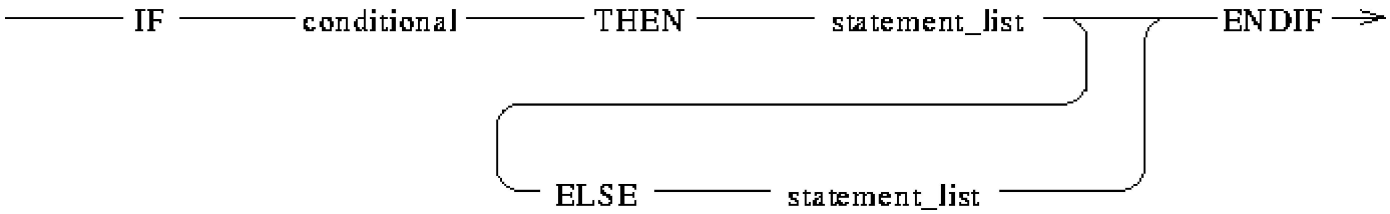


statement :



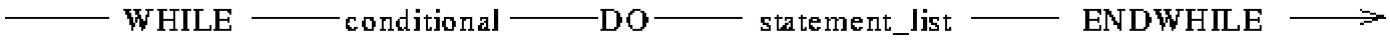
assignment_statement :

if_statement :

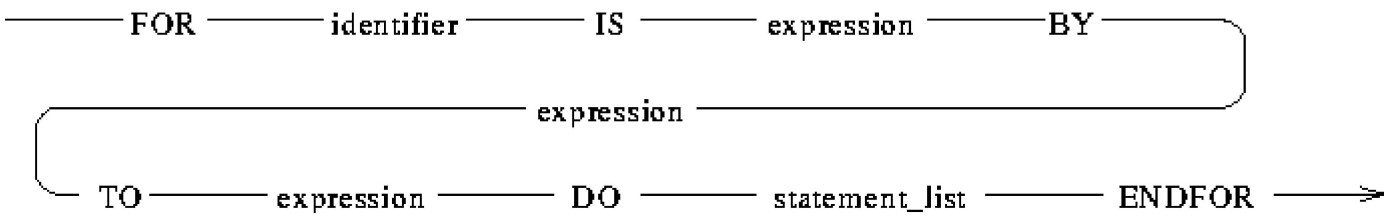


do_statement :

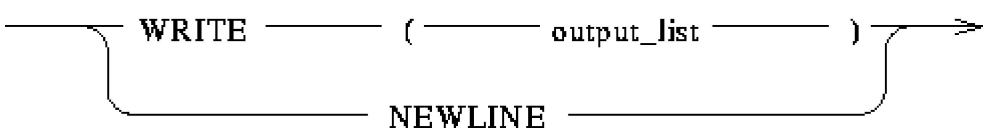
while_statement :



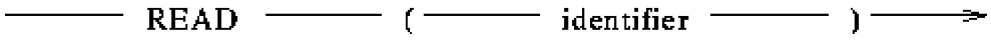
for_statement :



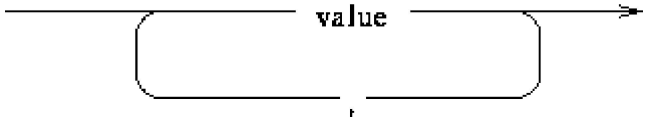
write_statement :



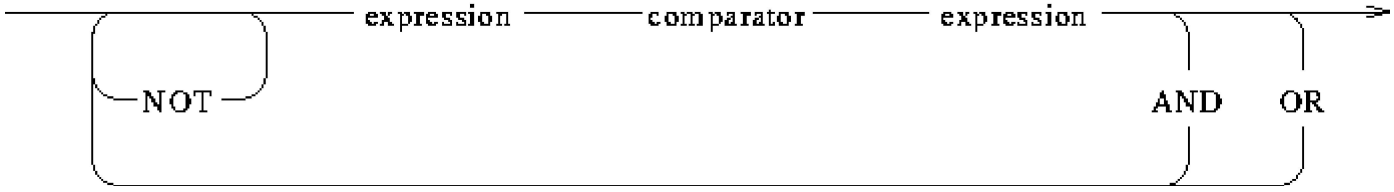
read_statement :



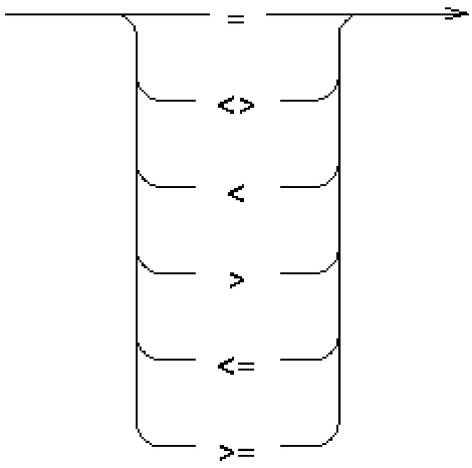
output_list :



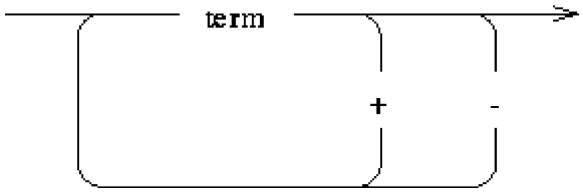
conditional :



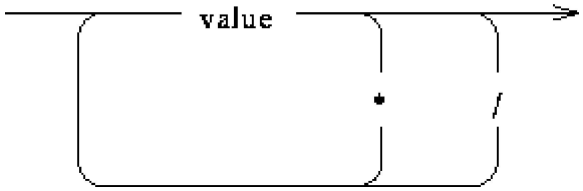
comparator :



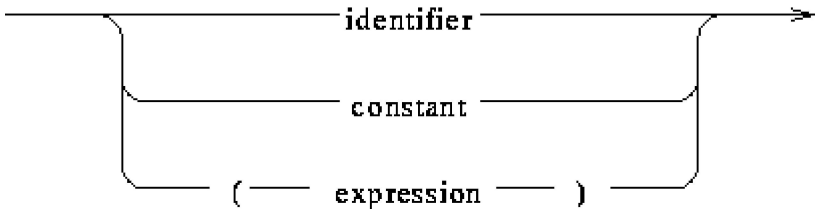
expression :



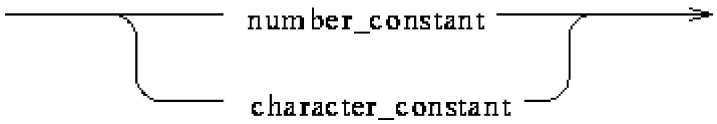
term :



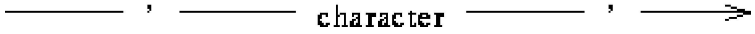
value :



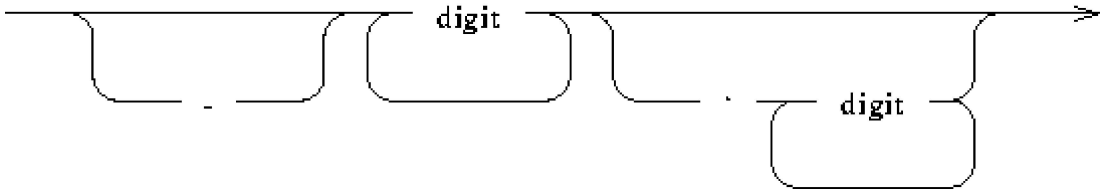
constant :



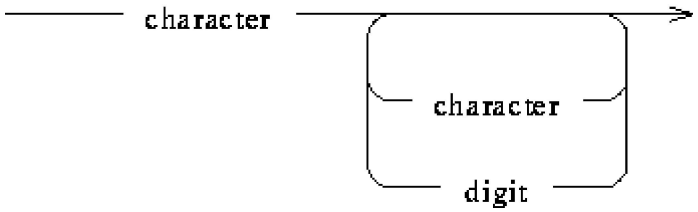
character_constant :



number_constant :



identifier :



Characters can be any upper or lower case letter.

Digits are any from 0 to 9.