Input and output are (desired) side effects needed in practical prog. languages.

Up to now:

- **input**: arguments for pred. symbols
- **output**: answer subst.

But Prolog also has built-in predicates for "proper" input and output.

**Pred**: write/1

Proof of write(T) succeeds, as a side-effect, the term T is written to the current output stream. (Default output stream: screen).

?- X is 2+3, write(X).
5 <- printed on screen
X=5 <- resubstituting answer subst.

**mult**(X,Y) :- Result is X * Y, write(X * Y),
write(’ = ’), write(Result).

?- mult(3,4).
3 * 4 = 12
Result = 12
Side-effects of Predicates are not undone when backtracking in the SLD-Evals:

\[ q(a). \]
\[ q(b). \]
\[ p :- q(X), \text{write}(X), X=b. \]

\[ \text{?-p.} \]
\[ \text{ab} \]

There is also a predicate \text{nl}/0 for "new line":

\[ \text{?-write(a), nl, write(b), nl, write(c).} \]
\[ \text{a} \]
\[ \text{b} \]
\[ \text{c} \]

Pre-defined predicate \text{read/1} for input:

\text{read(L)} \quad \text{reads a term S from the current input stream (default: keyboard)}
the term is marked with.

If this does not succeed, then read (t) fails.

**Ex:** IO program computing squares.

One can also do input/output with files.
For this, one has to change the input/output stream.

- `see/1` changes the input resp.
- `tell/1` output stream
- `seen/0` sets input/output stream
- `told/0` back

When reading, reaches the end of a file, then read (X) results in the answer subst.

\[ X = \text{end-of-file}. \]

**Ex:** input file could contain

3. -4.