Statements

Expressions vs. statements
==========

Algol68
```
x := begin int i; read(i); i+2 end;
x := ( int i; read(i); i+2 );
x := if y > z then y else z fi;
```

Abstract specification:
- Textual
- Flow charts
  - Structograms (Nassie-Schneider diagrams)
  -

Commands
(role of ; == separator vs terminator)
```
pascal:   begin x := 1  ;  y := 2 ; end
          if x > y then x := 1 else y := 1 end
C/C++:    if (x > y) x = 1; else y = 1;
```

skip, null-statement
compound statement, i.e. "block"

assignment (max be expression, in C/C++)
```
var := expr  (pascal)
*ptr = *qtr;
```
ADA := is statement "private limited" -> assignment is forbidden

multiple assignment   x := y := z   x := z and y := z
C/C++   x = y = z   x = (y = z)
C++17 structured binding   [x, y] = [1, 3.14]

function call (max be expression, in C/C++)

```
struct C  // Functor
{
  int operator()(int) { ... }
};
```

```
C c;  c(3) --> c.operator()(3)
Matrix m;      m(3,4)  -> m.at(3,4)
```

branch statements
==========
if - elif - else
switch break/default
(pattern matching) (SCALA)

loops
====
while
do-while
for
for-each for (auto i : v)
break/continue

goto

return