Code without loops or recursion

Rule: It takes constant time, written "$O(1)$".

Example

```
for i : 100 .. n
    sum := sum + 1
end for
```

Example

```
var sum := 0
for i: 1 .. n div 2
    for j: 1 .. n*n
        sum := sum + 1
    end for
end for
```

Analysis:

- Each iteration of the inner loop takes $O(1)$ time.
- On every iteration of the outer loop, $O(n^2)$ iterations of the inner loop are performed.
- Thus each iteration of the outer loop takes $O(n^2)$ time.
- The outer loop is performed $\lfloor n/2 \rfloor$ times and $\lfloor n/2 \rfloor$ is $O(n)$.
- Therefore the loop takes $O(n^3)$ time.
- Therefore the program takes $O(n^3 + 1)$ time. (1 is for the initialization.)
- Thus the entire program takes $O(n^3)$ time.

Write your analyses in this style. Annotating the code is not sufficient.

Loops

loop

$\vdots$

end loop

for $<$ *range* $>$

$\vdots$

end for

RULE: If the body of a loop is executed $O(f(n))$ times and each iteration takes $O(g(n))$ time, then the entire for loop takes

$$O(1 + f(n) \cdot g(n))$$

time.