**Aufgabenblatt 3 - O-Notation**

**Aufgabe 1: Komplexitätsklassen**

Füllen Sie die folgende Tabelle mit den unterschiedlichen Funktionswerten für die angegebenen Werte für n aus,

mindestens so weit Ihr Taschenrechner reicht. Welche Schlussfolgerungen können Sie aus den Werten in der Tabelle auf

die Laufzeit von Algorithmen ziehen?

| N | log\_2 (n) | √n | n\* log\_2 (n) | n^2 | n^3 | 2^n | 3^n |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **16** | 4 | 4 | 64 | 256 | 4096 | 65536 | 43046721 |
| **64** | 6 | 8 | 384 | 4096 | 262144 | 18446744073709551616 | 3433683820292512484657849089281 |
| **256** | 8 | 16 | 2048 | 65536 | 16777216 | 115792089237316195423570985008687907853269984665640564039457584007913129639936 | 139008452377144732764939786789661303114218850808529137991604  824430036072629766435941001769154109609521811665540548899435521 |
| **1024** | 10 | 32 | 10240 | 1048576 | 1073741824 | 179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789758814416622492847430639474124377767893424865485276302219601246094119453082952085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216 | 373391848741020043532959754184866588225409776783734007750636931722079040617265251229993688938803977220468765065431475158108727054592160858581351336982809187314191748594262580938807019951956404285571818041046681288797402925517658012340617298396574731619152386723046235125934896058590588284654793540505936202376547807442730582144527058988756251452817793413352141920744623027518729185432862375737063985485319476416926263819972887006907013899256524297198527698749274196276811060702333710356481 |
| 2^20 | 20 | 1024 | 20971520 | 1099511627776 | 1152921504606846976 | ∞ | ∞ |

Die Laufzeit des Algorithmus ist eine Kombination aus linearem und logarithmischen Aufwand und steigt etwas stärker als linear mit der Eingabegröße an. Moderateraufwand

Algorithmen mit logarithmischem Zeitaufwand (O(log n)) sind auch für große *n* effizient.

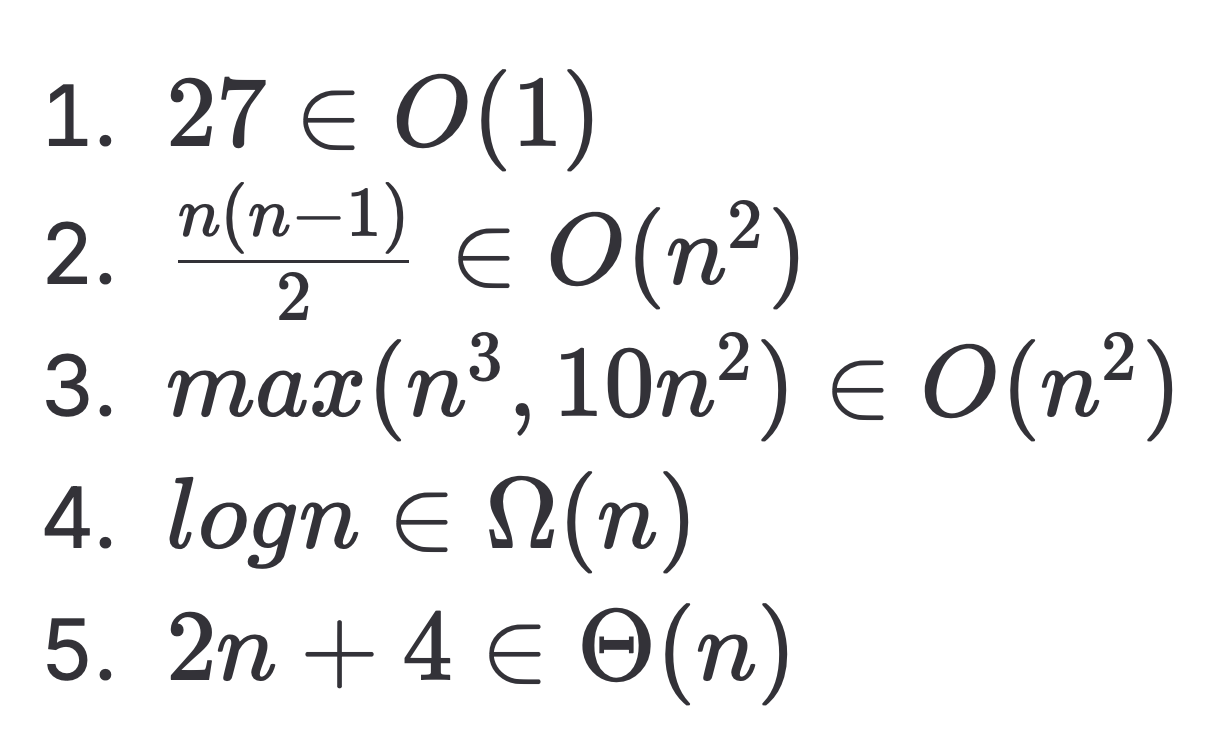
Algorithmen mit quadratischem Zeitaufwand (O(n^2)) sind für moderate *n* noch anwendbar, aber sie werden schnell unpraktikabel, wenn *n* wächst.

Algorithmen mit kubischem Zeitaufwand (O(n^3)) sind bereits für relativ kleine *n* ineffizient.

Exponentielle Algorithmen (O(2^n)) und (O(3^n)) sind extrem ineffizient und praktisch nicht anwendbar, sogar für kleine n.

**Aufgabe 2: O-Notation**

Zeigen Sie, dass die folgenden Aussagen wahr sind oder widerlegen Sie sie. Begründen Sie Ihre Entscheidung und geben Sie

bei wahren Aussagen ein geeignetes c und n0 an.

<https://share.goodnotes.com/s/6nYE4EdnefcNgm0Z3eDaBH>

**1.**

27 ist eine Konstante daher war

**2.**

| \*2

| c= 1/2

gilt wenn und c = 1/2

Wahr da c und positiv sind

**3.**

Wahr

**4.**

|  |  |
| --- | --- |
| N |  |
| 1 | 0 |
| 2 | 1 |
| 3 | 1,58496 |
| 4 | 2 |
| 4000 | 11,9658 |

nicht wahr da Bedingung nicht erfüllt

**5.**

| -4

| c = 3

Wahr

| c = 1

Wahr

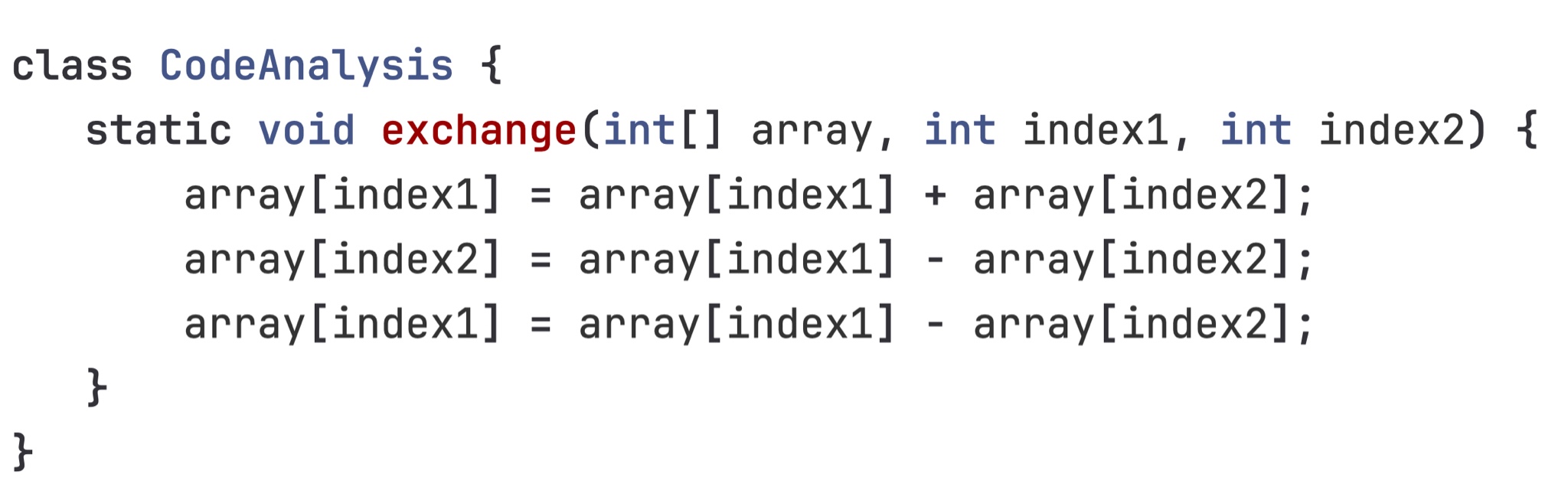
= Wahr

**Aufgabe 3: Codeanalyse**

Bestimmen Sie zunächst die Anzahl elementarer Rechenschritte wie Vergleiche, Zuweisungen und arithmetischer Operationen

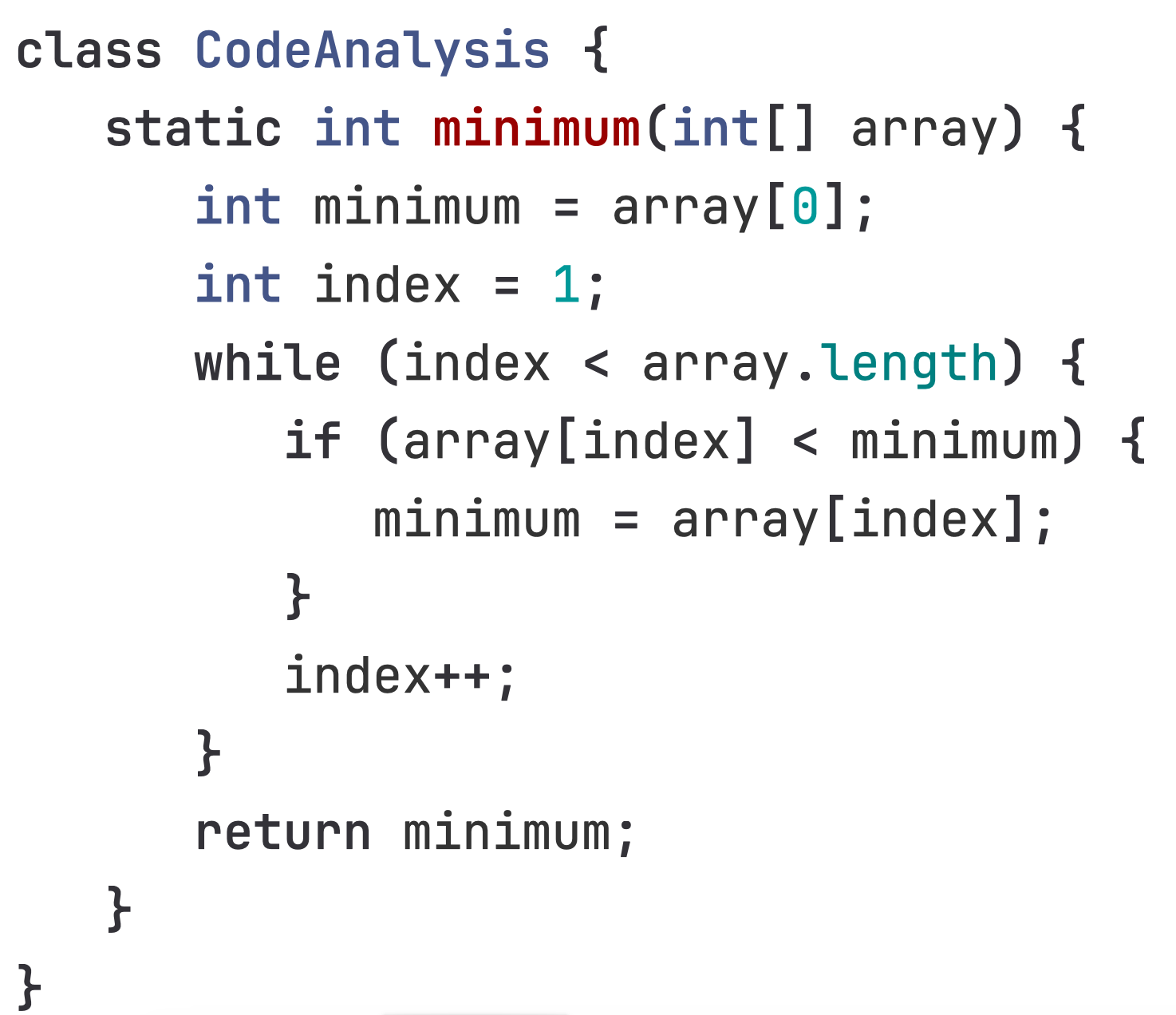
für die folgenden Codestücke in Abhängigkeit von der Anzahl n „beteiligter“ Elemente. Die zu betrachtenden

Elementarschritte sind jeweils angegeben. Geben Sie dann den Aufwand in O-Notation an.

1. Vertauschen zweier Feldelemente: (Anzahl der Zuweisungen?)Rechteck RechteckRechteck RechteckRechteck RechteckRechteck RechteckRechteck RechteckRechteck Rechteck

Zuweisungen = 3

Anzahl elementarer Rechenschritte = 6

2. Suche des Minimums in einem Array: (Anzahl der Vergleiche?)

Rechteck Rechteck

Rechteck Rechteck

Rechteck Rechteck

Rechteck Rechteck

Rechteck Rechteck

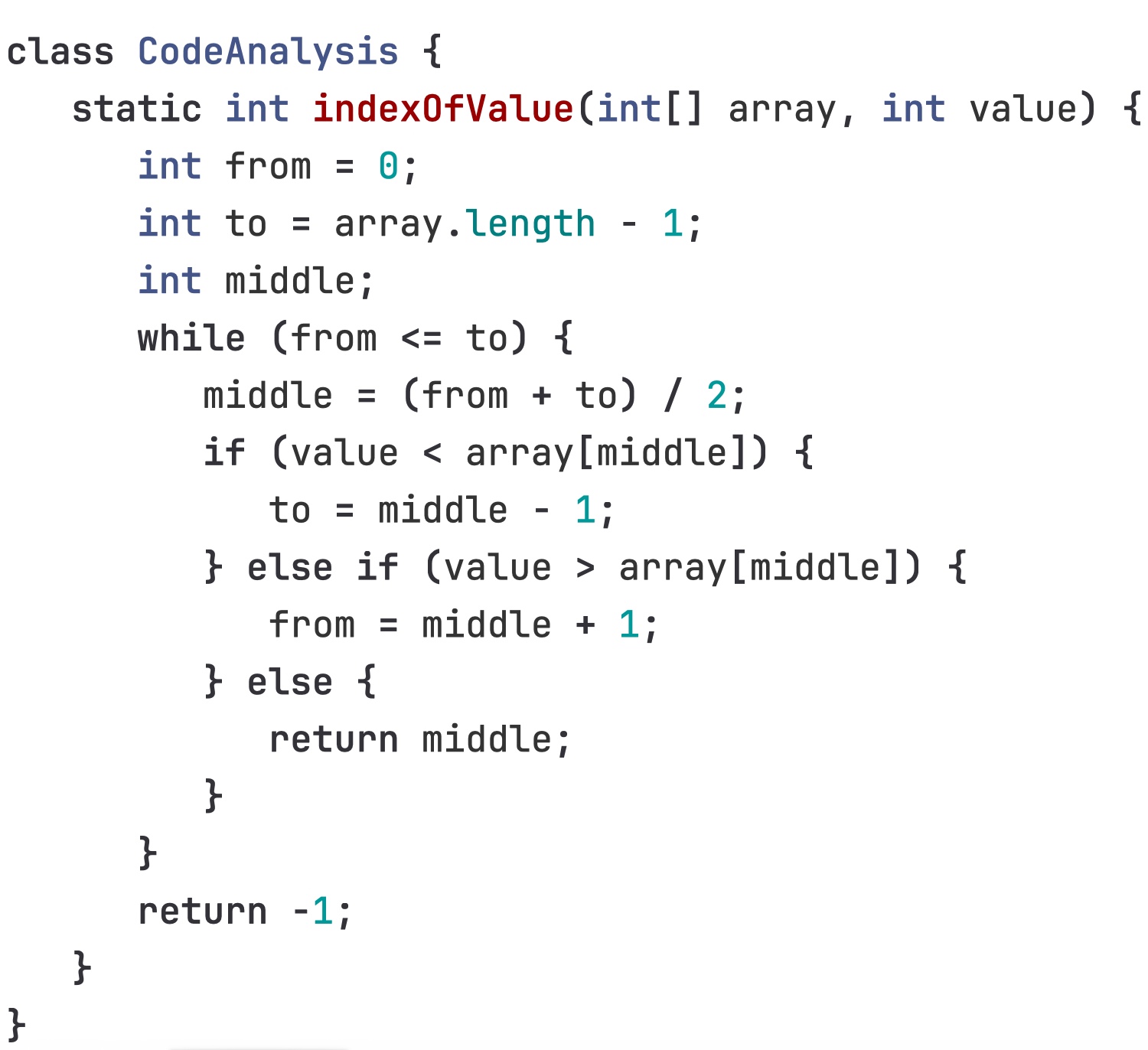
Rechteck Rechteck

Zuweisungen = 3

Vergleiche = 2

Anzahl elementarer Rechenschritte = 6

3. Suche eines Wertes in einem sortierten Array: (Anzahl der Vergleiche?)



Rechteck Rechteck

Rechteck RechteckRechteck Rechteck

Rechteck Rechteck

Rechteck RechteckRechteck RechteckRechteck Rechteck

Rechteck Rechteck

Rechteck RechteckRechteck Rechteck

Rechteck Rechteck

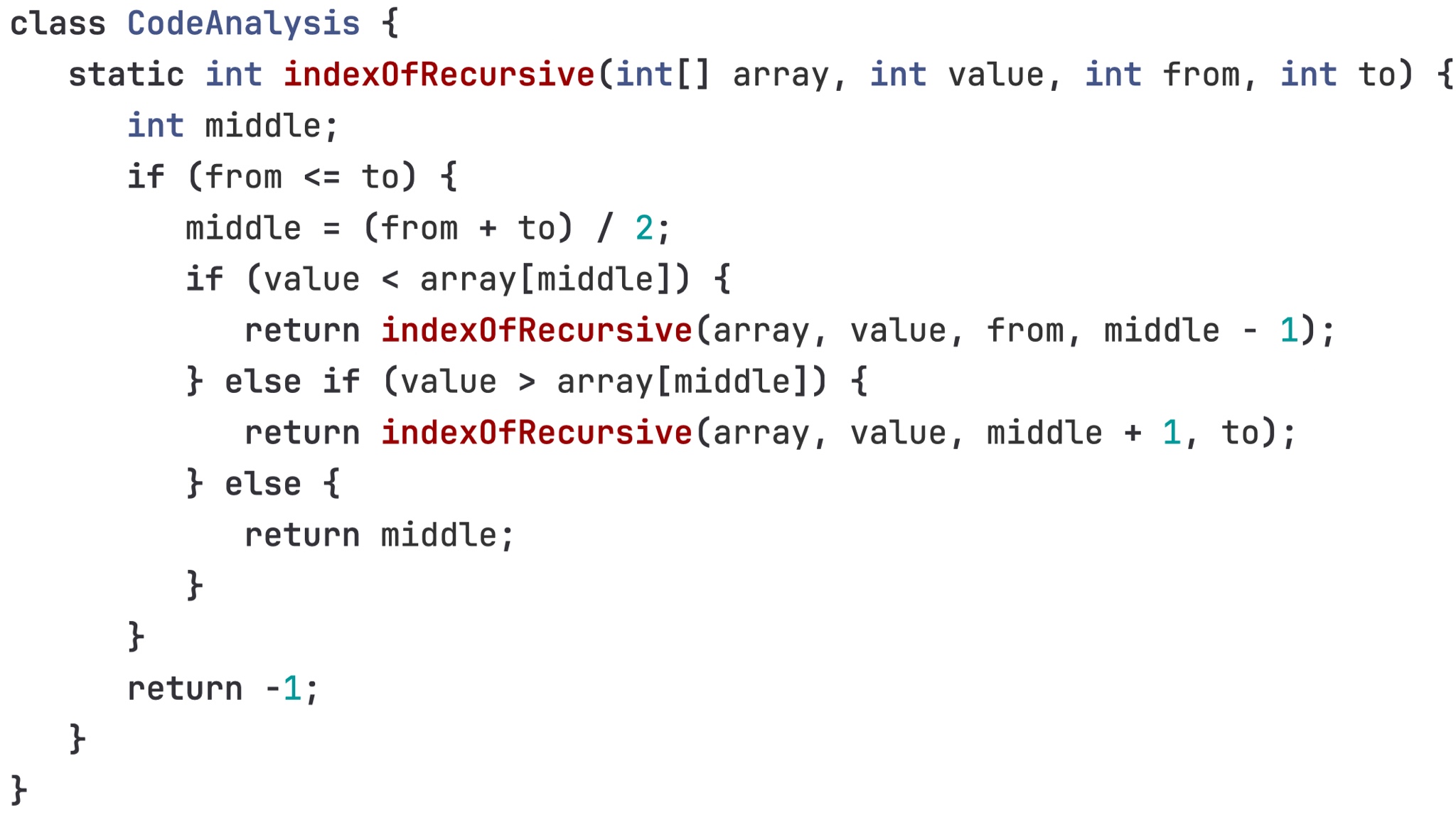
Rechteck RechteckRechteck Rechteck

Zuweisungen = 5

Vergleiche = 3

Anzahl elementarer Rechenschritte = 13

**Aufgabe 4: O-Notation**

Die folgende Funktion implementiert die Suche eines Wertes in einem sortierten Array. Bestimmen Sie zunächst die rekursive Aufwandsfunktion für die Anzahl der erforderlichen Vergleiche und bestimmen Sie dann die Komplexität der Funktion in O-Notation.Rechteck RechteckRechteck RechteckRechteck RechteckRechteck RechteckRechteck RechteckRechteck RechteckRechteck Rechteck

|  |  |  |
| --- | --- | --- |
| N | t(n) | Log n |
| 1 | 1 | 0 |
| 2 | 2 | 1 |
| 4 | 3 | 2 |
| 8 | 4 | 3 |
| 16 | 5 | 4 |
| 32 | 6 | 5 |
| 64 | 7 | 6 |

Rechteck Rechteck

Rechteck Rechteck

Rechteck Rechteck

a = 1

b = 2

f(n) = 1